INSMUS-8

INSNOTRAN MUSIC COMPILER

INSPLAY 4-VOICE INSTRUMENT SYNTHESIS INTERPRETER

21 ENTRY INSTRUMENT LIBRARY

FOR MTU-130/140 COMPUTERS

MARCH, 1984 Rev. A

DISCALIMER OF ALL WARRANTIES AND LIABILITY

No warranties, either express or implied, are made by Micro Technology Unlimited with respect to this manual or the software described herein, its quality, performance, merchantability, or fitness for any particular application. This product is sold "as is". The buyer assumes all risk as to quality and performance. Under no circumstances will MTU be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software, even if MTU has been advised of the possibility of such damages. Should the software prove defective following purchase, the buyer assumes the entire cost of all necessary servicing, repair, or correction and any incidental, indirect, or consequential damages. Additional rights vary from state to state so some of the above exclusions and limitations may not apply to you.

Micro Technology Unlimited reserves the right to make changes to the product and specifications described in this manual at any time without notice.

ACKNOWLEDGEMENTS

INSMUS-8 is just the lastest in a whole series of D-to-A converter music packages for 6502 based computers. The first was released in 1977 for the KIM-1 and was followed by versions for the PET, Apple II, and MTU-130/140.

This manual, the programs, and the encoded songs in this package are the result of efforts by numerous people both inside and outside MTU over a period of several years. The most instrumental of these are listed below:

Cliff Ashcraft - Instrument definition design, song coding, temperament feature.

Greg Byrd, MTU - INSNOTRAN, instrument library.

Hal Chamberlin, MTU - Idea development, INSNOTRAN, "polishing", manual editor.

Frank Covitz - Idea development, INSPLAY, coding of many songs.

TABLE OF CONTENTS

1. INTRODUCTION

INSMUS-8 is a pair of programs that combines the convenience of score entry allowed by the standard SNOTRAN music compiler with the sound synthesis flexibility of the Instrument Synthesis music system. The INSNOTRAN compiler translates a musical score expressed as English words and combinations of letters and numbers into a hexadecimal "pseudo code" that efficiently represents the musical and acoustical information in the score. The INSPLAY interpreter quickly recognizes the pseudo code and directs a sophisticated "software synthesizer" to generate the desired sound. Compared to the standard SNOTRAN and SPLAY programs provided with all MTU-130/140s, the INSNOTRAN/INSPLAY system provides much greater variety and realism in the reproduced sound and is much easier to understand and use than the older INSMUS system. In particular, the need for understanding machine language programming concepts, such as hexadecimal arithmetic and computer memory organization, is minimized, while flexibility and ease of entry are greatly enhanced.

Besides ease of use, several advanced features have been added that were not present in earlier music systems. The temperament (tuning) of notes may be controlled and note durations may be locally modified in the score for greater expressiveness. Automatic amplitude optimization of instrument definitions is now provided and an instrument library facility is available. During playing, the music may be stopped, backed up, replayed, and edited on the spot. The contents of waveform tables may also be displayed on demand and a "live keyboard" mode allows experimentation.

The following chapters will provide a complete description of the INSMUS-8 system along with examples and suggestions for effective use. Appendix A contains a complete example score and Appendix F gives a glossary of words used in this manual that may be unfamiliar. For readers interested in greater detail, MTU has several additional publications available on the subject of computer music. These are listed along with other several others in Appendix G.

1.1 MUSIC SYNTHESIS ON THE MTU-130/140

Music (and other sound) is synthesized on the MTU-130/140 with a hardware device called a "digital-to-analog converter" or DAC for short. Quite simply put, a DAC is capable of transforming a string of numbers representing the actual shape of sound waves into an electrical signal that will indeed create that sound wave shape when played through a loudspeaker. This is accomplished much like one would plot a smoothly varying function on a graph -- with a lot of closely spaced points. For sound of reasonable quality, at least 8,000 of these points or "samples" must be fed to the DAC every second. Accomplishing this requires a good deal of highly efficient machine language programming, which is part of the INSPLAY program. The ISNOTRAN Compiler translates your instructions written in a music-oriented language into sound synthesis instructions for the INSPLAY program.

Most other computers use a hardware "synthesizer chip" to create musical sounds. When using such chips, the tonal variety and number of simultaneous sounds that may be synthesized is fixed in hardware. Often the only tone color available is a square wave which sounds like a kazoo, and the maximum number of simultaneous notes is limited to 3 or 4. The digital-to-analog converter on the MTU-130/140, on the other hand, has no such hardware imposed limitations since software actually computes the sound waveforms. In essence, the INSPLAY program simulates what a synthesizer chip would do. There are microprocessor speed limitations, to be sure, but the system described in this manual has more capability than any general purpose computer using a synthesizer chip currently available.

1.2 INSNOTRAN COMPILER

Communicating a musical score to a computer presents a difficult problem to the system designer because of musical notation's pictorial and often ambiguous nature. One effective way to input music is through the use of a MUSIC LANGUAGE in which the note pitches and durations are described with letters and numbers. When encoded in this way, the score can then be entered and edited with a standard text editor, rather than a specialized music editor.

Over the years, many "alphanumeric" music languages have been developed for different purposes. NOTRAN (NOte TRANslator) is one such language developed by Hal Chamberlin in 1970 which is exceptionally easy to learn and read. It is best suited for encoding "conventional" music scores. The INstrument Synthesis NOTRAN compiler (INSNOTRAN) described here accepts a subset of the NOTRAN language. Its purpose is to translate an alphanumeric score written in that subset into instructions for the companion INSPLAY sound synthesis program.

In operation, the user first prepares an INSNOTRAN score file using a text editor or perhaps a music composition program. The score exists in this form as a standard text file on disk that may be displayed, printed, or edited further at will. Next, the INSNOTRAN compiler program is run. The compiler will ask for the name of the score file to be compiled. It will also ask for an "object file" name which will contain the detail coded sound synthesis instructions. The compiler then reads the score file, interprets the meaning of the INSNOTRAN statements, and writes the sound synthesis instructions onto the object file. At the same time, a listing showing the INSNOTRAN score, corresponding object data, and any errors detected can be printed or displayed or saved to a "listing file". To hear the actual music, the object file just created is loaded into memory and the INSPLAY program is run. This program follows the compiled sysnthesis instructions to produce the sounds specified by the original INSNOTRAN score.

1.3 INSPLAY INTERPRETER

INSPLAY is a program that interprets compactly coded synthesis instrucions stored in memory and efficiently computes the corresponding sound waveform points. As each point is computed, it is written into a register in the 8-bit audio digital-to-analog converter that is part of the MTU-130/140's music system. These points are computed and written at the rate of 8,772 points per second, giving a frequency response from below 20 to about 3,500 Hz. This is certainly not hi-fi, but is comparable to AM radio reception through a typical receiver.

The INSPLAY program simulates a 4-voice sound synthesizer. Sounds are created using a waveform table look-up process. Many slightly different waveforms are computed based on a user-specified harmonic mixture and are stored as tables in memory. Amplitude envelopes are generated by the use of "waveform sequence tables," which specify which waveforms are to be played and when. Thus realistic sounding "instruments" can be defined. Each voice can be assigned to a different instrument, and there may be up to 15 instruments "on-line" in memory to choose from. If a desired instrument is not in memory when needed, it can be defined in a matter of seconds. The INSPLAY system is flexible and easy to use, especially when accessed through the INSNOTRAN compiler.

Besides playing the the compiled music score, INSPLAY allows a considerable amount of "real time" control over the playing process. Function keys with displayed screen legends make selecting the desired control option simple. At any time the music playing may be halted ("pause" mode) and restarted. When in pause mode, the last played "event" (note or chord) may be replayed at will or

the music may be played backward until a previously heard wrong note is reached. The current event may also be edited directly in memory to "patch" minor mistakes that can be later fixed permanently in the text score.

Also while in pause mode, the actual waveforms of selected instruments may be displayed on the screen. Finally, the MTU-130/140 keyboard can function as a music keyboard with a 3 octave range. Although the keyboard mode only allows monophonic (one note at a time) playing, it is useful for exploring the full range of an instrument's sound without having to write a "test" score.

1.4 PLAYING AN EXISTING, COMPILED SCORE

This section will briefly describe how to play a song score that has already been compiled. It will describe only the most basic operation of the INSPLAY program. For additional information about INSPLAY, see Chapter 3.

The format of the INSPLAY command is:

INSPLAY [<musicobjectfile>] ...

where <musicobjectfile> is the name of a previously compiled music object file. If desired, an extension and drive number may be appended to the filename. If more than one filename is given, the files will be loaded and played in sequence. If no filename is given, a compiled score is assumed to already exist in memory starting at address $2000.

For example, to play the file SAMPLE.M (which is included on the INSMUS-8 distribution disk), enter the following command in response to a CODOS> prompt (disk with INSPLAY and SAMPLE.M assumed to be on disk in drive 0):

INSPLAY SAMPLE.M

After a small amount of disk activity to load INSPLAY and the SAMPLE.M object file, the screen will be cleared, INSPLAY will identify itself, and the score in memory will be interpreted.

For most songs, a number of waveform tables must be computed before music can be played. INSPLAY will display the progress of these computations by identifying the waveform set, harmonic number, and instrument ID being processed. Unless the instrument definitions are very complex, this process will require from a few seconds to less than a minute. If the score calls for it, the shape of computed waveforms and envelopes will also be displayed. In this case, the f4 function key legend (labelled "PAUSE") will light up and INSPLAY will wait until f4 is pressed before continuing.

After the waveform tables are computed, the song will begin to play. It is normal for the average volume level to be somewhat less than that produced by the standard SPLAY interpreter. If necessary, the MTU-130/140's volume control can be adjusted. You may temporarily stop the music by pressing the PAUSE function key (f4) or abort it altogether by pressing ABORT (f1). When in pause mode (f4 legend lit in reverse video), several other functions are available which are more fully described in Chapter 3. The PAUSE key may be pressed again to leave pause mode and resume playing. When the song is complete (or ABORT is pressed), INSPLAY will return to CODOS unless there was more than one filename specified in the INSPLAY command in which case the next song will be loaded and played.

1.5 COMPILING AN EXISTING TEXT SCORE

This section will describe how to compile a song score in text form that has been composed or transcribed using a text editor. The result of the compilation is a music object file and a listing file. The object file may then be played by INSPLAY (see section 1.4 above) and the listing file printed if desired. Additional information about the INSNOTRAN music compiler may be found in Chapter 2.

To start the INSNOTRAN compiler, enter the following command in response to a CODOS> prompt (INSNOTRAN assumed to be on the disk in drive 0):

INSNOTRAN

INSNOTRAN will be loaded then it will clear the screen, identify itself, and ask:

ENTER NAME OF SOURCE FILE -

Enter the name of the text file that contains the text score you wish to compile. Be sure to append a drive number if it is not on the disk in drive 0.

INSNOTRAN will next ask:

ENTER LISTING FILE NAME OR DEVICE NAME (N or CR=NONE) -

If you wish to have a listing file generated for later printing or examination, enter a legal filename. If that name already exists (perhaps from the last time this score was compiled), INSNOTRAN will ask if it is alright to overwrite the old file. If you have a printer connected and wish an immediate printed listing, you may simply enter P. Entering C will cause the listing to be displayed on the screen. An N entry will suppress the listing file. If the listing is being directed to a file and you have adopted a naming convention in which the listing file name is the same as the source file name except for the extension, you may simply enter cntl/B to recall the source file name entered earlier, backspace once to the extension letter, type the extension letter for the listing file, and type return to continue.

INSNOTRAN will next ask:

PLEASE ENTER OBJECT FILE NAME (N=NONE) -

Enter a legal file name for the object file. It is suggested that a .M extension be used to identify the file as a music object file. If the name already exists, INSNOTRAN will ask if it is all right to overwrite the old file. Entering N will suppress the object file but the compilation will still be performed and listing file generated. As with the listing file, you may use cntl/B to speed entry of the object file name.

The text score will now be compiled and any errors encountered will be displayed and inserted into the listing file. Note that it may take a substantial amount of time to compile instrument definition statements (due to the amplitude optimization algorithm) so if the listing is being directed to a file, there may be long periods of apparent system inactivity. Such statements are typically first in the score. Also note that the listing file is internally "buffered". This means that if the listing is directed to the screen or printer, nothing will be output until about 1000 characters have been accumulated from the compilation. When the buffer is full, its contents will be output all at once in a 1000 character chunk. While the effect may seem strange, all is normal and no data will be lost.

Following compilation, INSNOTRAN will display *** COMPILATION COMPLETE *** and give a count of the number of errors encountered. See Appendix B for an explanation of INSNOTRAN error messages. Control is then returned to CODOS.

The process of recompiling and playing a score can be simplified by preparing a job file with the necessary commands. For example, the job file:

```
INSNOTRAN          ; Start INSNOTRAN compiler
SAMPLE.T           ; Name of score file to be compiled
SAMPLE.L           ; Name of listing file
Y                  ; Answer to "OK TO OVERWRITE?"
SAMPLE.M           ; Name of object file
Y                  ; Answer to "OK TO OVERWRITE?"
INSPLAY SAMPLE.M ; Play the resulting object file
```

would start the compilation/play sequence for the example score on the INSMUS-8 distribution disk. Note that for this job file to work, the compile/play cycle must have already been performed once manually so that the SAMPLE.L and SAMPLE.M files already exist.

2. THE INSNOTRAN MUSIC COMPILER

Translating a written music score into INSNOTRAN statements is a simple, almost mechanical process that is easy to master. There is much room for creativity, however, in the orchestration and arrangement of the piece. Or if you have musical experience, the notes themselves can be altered to suit your taste or entirely new compositions may be written.

An INSNOTRAN score file is simply a text file consisting of lines of ASCII text. The score file may be created and edited using the standard MTU-130/140 text editor, the WORDPIC word processor, or any other kind of available editor. Conceivably a user written composition program written in BASIC or some other language could also generate or modify a score file.

2.1 STRUCTURE OF AN INSNOTRAN SCORE

An INSNOTRAN score consists of two sections, the Command Section and the Notes Section. Statements in the Commands Section specify such things as the tempo, number of voices, instrument definitions and assignments, and the playing sequence of segments of the score including repeats. Statements in the Notes Section specify the actual notes to be played.

Typically the score is broken into a number of "segments" where each segment is a group of related notes. By means of the commands in the Commands Section, the tempo, voice assignments, and other parameters may be changed between segments. Since most music contains a great deal of repitition, only the segments that are unique need be coded in the Notes Section and commands in the Commands Section can be used to play the segments in proper order.

The structure of a typical INSNOTRAN score can be seen by examining the skeleton below:

```
                *-comments-
                NVOICES 4
                NEWINS 2 ...
                NEWINS 3 ...
Commands        ASSIGN 2,2,3,3
Section         TEMPO 1/4=500
                PLAY 100
                PLAY 200
                TEMPO 1/4=400
                PLAY 100
                ENDCMD
--------        *-comments-
                SEGMENT 100
                 -notes-
                 -notes-
Notes           ENDSEG
Section         *-comments-
                SEGMENT 200
                 -notes-
                 -notes-
                ENDSEG
                END
```

The initial -comments- typically give the song's title, the coder's name, and other identifying information. The NVOICES, NEWINS, and ASSIGN statements define characteristics of the "orchestra" to be used. The TEMPO statement establishes the speed at which the music is to be played. The PLAY 100

statement plays the first segment of notes, and the PLAY 200 statement plays the second segment. Then the tempo is increased slightly and finally the first segment of notes is played again. The ENDCMD statement marks the end of the Commands Section and thus ends the performance.

In the Notes Section is coding for the first segment (arbitrarily called segment number 100), and then coding for the second segment (200). Comments preface each segment identifying to the human reader what part of the score is included in the segment. Each segment starts with a SEGMENT statement, contains the notes associated with it, and ends with an ENDSEG statement. An END statement after the last segment marks the end of the INSNOTRAN score.

2.2 FORMAT OF INSNOTRAN STATEMENTS

An INSNOTRAN score is composed of lines of text where each line is generally an individual statement, much like any other programming language. The length of an INSNOTRAN line can be anything up to 254 characters, but you will probably want to make lines 80 columns or less to simplify editing. In fact there is an incentive to keep them down to 53 characters or less, because then the listing lines produced when the score is compiled will fit on an 80 column display or printer without being broken. All statements are one line long except possibly the NEWINS, WAVESET, QWAVESET, QSEQTAB, and TEMPRMNT statements, which can be continued to several lines.

Lines that begin with an * in column 1 are regarded as comment lines and are ignored by the compiler. Comment lines will however be printed on the listing. Completely blank lines are not allowed; if you want to space out the score for improved readability, put an * at the beginning of any blank lines. Lines beginning with anything other than an * are examined by the compiler and are expected to be a valid command or note statement.

INSNOTRAN statements are generally made up of keywords, keyletters, numbers, and punctuation. Keywords are command names such as TEMPO, or WAVESET, or END. Keywords must be spelled out in all capitals and must be one word, that is, not broken up by blanks. Keyletters such as C, H, M, and # are used for various purposes in INSNOTRAN statements, depending on the statement being discussed. For example, in a note statement, C would denote a musical pitch, while in a WAVESET statement, H would refer to a harmonic. When a keyletter is called for, the exact upper case letter should be used.

Numbers are used extensively to define quantities such as tempo, octaves, percentages, and identifiers. In INSNOTRAN only whole numbers are used; a decimal point is flagged as an error. In cases where positive or negative numbers are required, the plus sign is optional for positive numbers; the minus sign is, of course, required for negative numbers. The largest number recognized is 65535, but most statements limit numbers to values much smaller than this. Commas must NOT be used in large numbers; instead, the digits must be strung together and there must be no intervening blanks. You may place blanks BEFORE a number, however, for improved readability.

Punctuation is used to separate the various elements of a statement from each other, such as numbers in a list. In most INSNOTRAN statements, blanks perform this function. In a couple of cases, however, commas, semicolons, or dashes are used to "tie together" a group of related parameters, while groups are separated by blanks. When blanks are used like this for punctuation, you may use any number you wish, as long as there is at least one.

Comments may be added at the end of a statement since the compiler never looks beyond the end of the meaningful part of a statement. Note however that some kinds of errors may indeed cause INSNOTRAN to look beyond the intended end of the statement and signal additional errors when it tries to interpret the comment.

Let's examine the anatomy of the INSNOTRAN statement below:

```
WAVESET 12    N=12  H1 0,200; 8,0;   H2 0,100; 8,0   BUILD WAVEFORM SET #2
```

It is not important to understand what the statement does right now, just the rationale behind its syntax. The word WAVESET at the beginning is a keyword and is also the name of the statement; thus this is called an WAVESET statement. When the compiler sees the keyword WAVESET, it knows how to interpret the rest of the information. Note that WAVESET is all the way to the left margin, as required. Following the keyword is a separating blank, then the first argument, which is the waveform set ID number. Then after several blanks comes the keyletter N followed by an = and a 12, meaning that there are 12 waveform tables in the set. Though several blanks were used to separate these arguments, only one was necessary.

Following the first two parameters are 2 PARAMETER GROUPS. The keyletter H signals the beginning of such a group. The number following the H indicates a specific harmonic number. The harmonic specification is followed by two pairs of numbers. Note that the numbers are separated by commas, and the PAIRS are separated by semicolons. The compiler continues to process the number pairs until there is no semicolon after the second number. It then assumes that all of the harmonics and number pairs of interest have been specified and goes to the next statement, ignoring the comment at the end of the statement.

Other statements are structured similarly although most of them are simpler than the WAVESET statement.

2.3 CODING A SAMPLE SCORE

In this section you will be led through the process of transcribing a sample score into INSNOTRAN. It is recommended that you read the discussion below in its entirety while referring to the musical score and its INSNOTRAN equivalent listed on the following pages. Points of interest are indicated by a circled letter. The score has already been typed in for you and may be found in the text file called EXAMPLE.T on the INSMUS-8 Distribution Disk. The listing produced when this score is compiled by INSNOTRAN may be found in appendix A.

2.3.1 Preliminaries

Referring to the sheet music in section 2.3.4 and the INSNOTRAN score in section 2.3.5, it is seen that the first three lines of the score are simply comment lines for identification. Next the name of the standard instrument library file is defined with a LIB statement. Following that, the number of voices is set to 4 since the vast majority of the music will use 4 (the maximum number) voices. Theoretically this is not necessary since 4 is the default but it doesn't hurt to be specific.

The next three statements use the instrument library to define instruments 1, 2, and 3. The W parameter (waveform set number) is each case has been made the same as the instrument number and is simply assigned in sequential order which is common practice. Presumably instrument 1 will sound like a plucked metal stringed instrument, instrument 2 will produce some kind of gargling

sound, and instrument number 3 will be a catgut stringed instrument.

The MODIFY statement following the instrument 3 definition actually modifies a part of instrument three's definition and calls the modification, instrument number 4. The modification creates a "repeat percussion" effect in which the repeat rate is the same as that of a string of thirty-second notes. Instrument 3 is not affected by this process.

The next two statements (WAVESET and SEQTAB) define a new instrument. The writer of this score apparently did not find an instrument sound to his liking in the library and so coded his own. Although the writer could have used a single NEWINS statement (which would have been simpler), he chose to use a WAVESET and SEQTAB pair (which does offer more flexibility). The WAVESET statement defines the instrument's sound in terms of harmonic envelopes for the fundamental through the fifth harmonic. The result of the statement is a set of 32 different waveforms in memory, each only slightly different from its neighbors. The "W5" specifies that it will be set number 5. The SEQTAB statement creates a special table that sequences the waveforms in a waveset in the proper order and at the proper speed for the desired sound. It refers to set number 5 (the "W5" again) and creates the sequence table for instrument number 5. It is the instrument number that will be referred to when notes are actually played. The other parameters define the instrument's overall envelope shape as a "simple stretch". After the new instrument is defined, a DRAWINS statement is included which will cause all of these waveforms to be drawn on the screen when the compiled score is played. This was probably inserted to aid in "debugging" the new instrument's sound.

The next two statements assign "voices", which are sound generators, to "instruments" whose sound has been defined above. It is common practice to use the lower numbered voices for the melody and higher pitched lines and the higher numbered voices for the bass lines. The first ASSIGN statement applies for playing mode 1 which is the default and used to play most notes. All 4 voices are assigned to instruments for mode 1. The second ASSIGN statement applies for playing mode 2. Apparently in this score, only voice 1 ever plays in mode 2 and none of the voices ever play in mode 3. By assigning voice 1 (which carries the melody in this score) to two different instruments via the modes feature, it is possible to switch from one instrument to the other on a note-to-note basis. As was seen above, instrument #3, which will play in mode 1, is the unmodified "Catstring" instrument while instrument #4 is the same except modified for a "strumming" effect. Presumably the writer will use mode 1 in the melody for the short notes and mode 2 for long notes in which strumming would be appropriate.

The last preliminary action is to specify a tempo. A look at the score (point A) shows that a quarter note is one beat and that the tempo should be 100 beats per minute. This implies that a quarter note is 1/100 of a minute or .6 second. Thus the appropriate tempo statement is *TEMPO 1/4=600* since .6 second is 600 milliseconds (thousanths of a second).

2.3.2 Segmenting the Score

Examination of the score reveals two rather long sections that are identical (points B-D is the same as points E-F). Therefore this part needs to be written in INSNOTRAN just once and then played twice. This repeated part is arbitrarily called Segment 10. Point G is another breaking point in the music and so a segment break is introduced there as well even though this point is not involved with repetition. Near the end of the score it is desired to gradually slow the tempo somewhat. This is accomplished by putting segment breaks at points I, J, and K. In the Commands Section, the tempo is redefined to be a little slower before each segment is played. Although somewhat awkward, attention to details

such as this makes the music sound much more natural.

Note that prior to playing Segment 20 (the PLAY 20 statement), that the pitch of voice 1 is raised by an octave. Since there were no OCTAVE statements prior to this, voice 1 (and 2-4 as well) all sound their normal pitch. After Segment 20 is played, another OCTAVE statement returns the pitch of voice 1 to normal.

2.3.3 Coding the Notes

Before coding the Notes Section (and indeed before coding anything), you should look over the score and decide upon a coding strategy. This includes voicing and segmentation decisions plus ways to "get around" difficulties that may be seen. Looking at the score for the example piece, one difficulty is immediately apparent; there are frequently more than 4 simultaneous notes. For coding into INSNOTRAN, some of these notes will have to be omitted. Examination of the score will reveal that the very lowest note is always (with just one exception) an octave lower than the next lower note. In addition, these are very low notes indeed, some as low as C1 which is about 33Hz and unlikely to be heard through the computer's speaker (or even most hi-fi systems). Thus these notes can be omitted which then allows about 95% of the score to be coded into 4 voices.

In the remaining cases, which are generally dramatic, sustained chords, some other notes will have to be omitted. The general rule here is that to maintain the richness intended by the songwriter, you should retain the lowest and highest notes written and omit any in between that duplicate retained notes at octave intervals. For example, at point C, 6 simultaneous notes are written. After omitting the very lowest as decided above, we are left with the following 5 pitches: F2, A3, C4, F4, and A4. The lowest (F2) and highest (A4) are retained and the A3 is dropped because it duplicates the A4. This is not a hard and fast rule however since at point J where 7 notes are written, it sounds better to retain the A3 and drop the F3 rather than vice-versa.

Rhythmically, this piece is very simple compared to more contemporary music. In fact, except for point H, it is strictly a series of chords. The coding at point H exactly parallels an example in section 2.5.7. More complex cases may require some thought or diagramming on graph paper and even experimentation but be assured that anything reasonable can indeed be expressed in INSNOTRAN.

Actually looking at the Notes Section, we see a comment line used as a visual separator followed by a MAXVOICE statement. Actually, since the MAXVOICE statement specifies 4, which is the default, it is not needed but it is best to be explicit. Next we see that the coder first recognized that all B's should be flatted thus a KEY statement to that effect was included. Note that the KEY statement identifies which note names should be automatically modified with a flat or sharp; it is <u>not</u> the "musical key" in which the song was written (this song is in the musical key of F). This convention, besides making more sense to non-musicians, allows for the use of non-standard scales. Since from now on all B's will be flatted, it is necessary to specify natural if a plain B is to be sounded. This is seen in line 11 in Segment 20 in which a tilda (~) was used after the note name to specify natural.

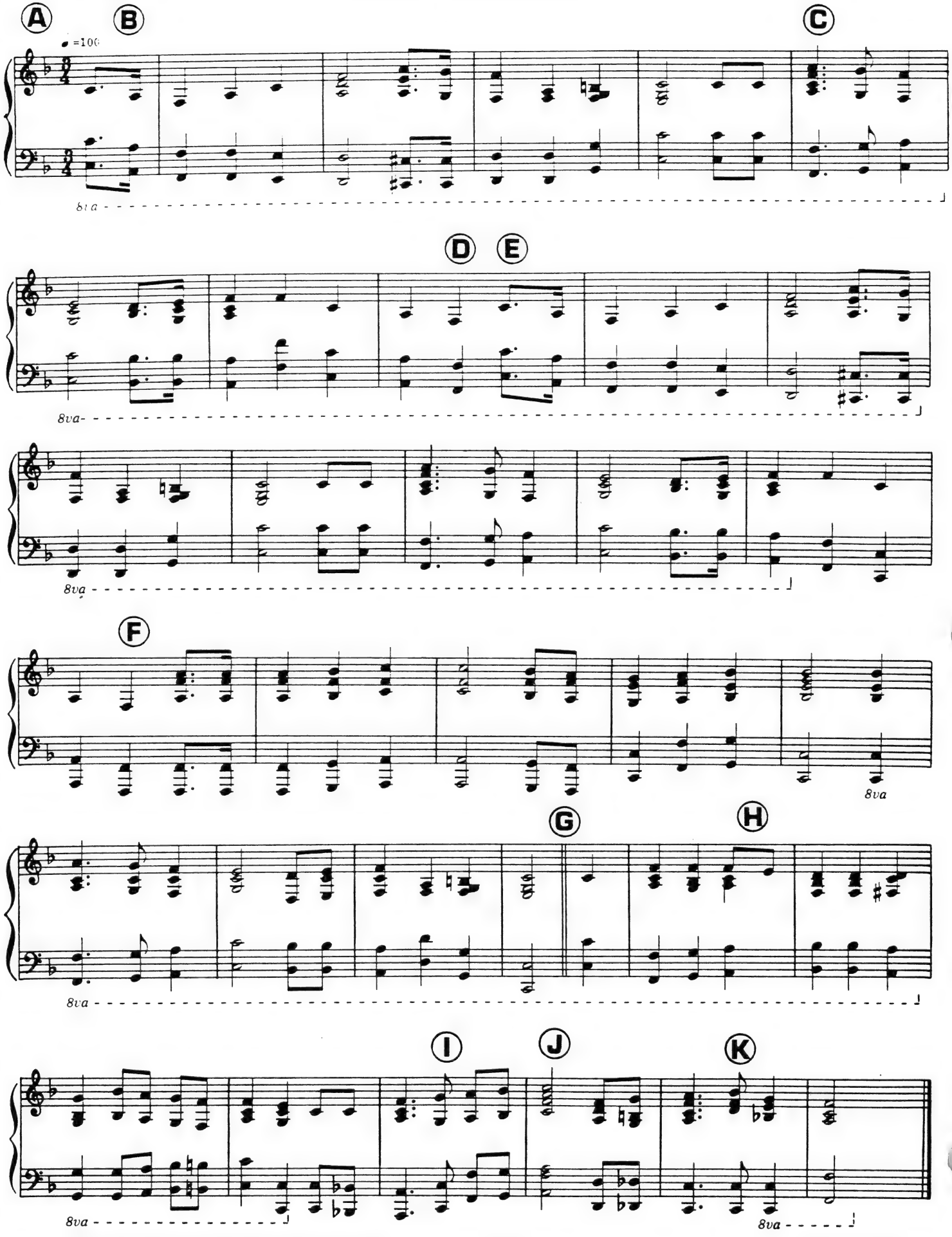
In segment 20, the modes feature is used on voice 1 to provide some variety. Since nothing was specified about modes prior to this in the Notes Section, all voices will play in mode 1. However, the first line in Segment 20 specifies mode 2 for voice 1. The very next line returns to mode 1 and then later another note is played in mode 2. The previously used mode always remains in effect until it is changed to a different mode. The first line in Segment 30 specifies

mode 1 which then persists until the end of the piece. It is important to note that the compiler goes through the Notes Section in sequential order even though the segments may be played in a different order than they appear. For this reason, it is best to keep the segments in order when possible.

Use of another feature is seen in segments 60 and 70. Here, one and then two leading + signs are used to lengthen the duration of the line they appear on. Each plus sign will lengthen the event by 12.5% which is about the minimum audible amount. Minus signs may be used to shorten an event, again by 12.5% for each. As many of either may be used as desired for accelerandos and retards the limit being that an event cannot be stretched beyond an unstretched whole note duration.

Many other features exist in the INSNOTRAN system which find use in different types of music. The following sections describe each of the statements in detail. Chapter 4 tells more about coding instrument definitions and Chapter 5 is devoted to a discussion of temperament. A more complex sample song is also included on the INSMUS-8 Distribution Disk under the name SAMPLE.T which can be used for further study of INSNOTRAN coding techniques.

Example Musical Score



```
* Example Score    Coded 8/3/82   G. Byrd
*
LIB INSTFILE.I
NVOICES 4
LIBINS 1 W1 METSTRNG
LIBINS 2 W2 GARGLE
LIBINS 3 W3 CATSTRNG
MODIFY 4=3 1/32
WAVESET W5 N=32 H1 0,0; 8,100; 31,0;
                H2 0,0; 2,40; 31,20;
                H3 0,0; 31,20;
                H4 0,0; 4,60; 31,0; H5 0,0; 24,20
SEQTAB 5 X=0 BR=05 W5
DRAWINS 5
ASSIGN  M1 V1=3; V2=5; V3=5; V4=1
ASSIGN  M2 V1=4
TEMPO 1/4=600    * POINT A
PLAY 10
PLAY 10
OCTAVE V1=1
PLAY 20
OCTAVE V1=0
PLAY 30
TEMPO 3/12=650
PLAY 40
TEMPO 1/4=725
PLAY 50
TEMPO 1/4=800
PLAY 60
TEMPO 1/4=1100
PLAY 70
ENDCMD
************************************************
MAXVOICE 4
KEY B@
SEGMENT 10
 BE 1C4,1/8.;                          4C3,1/8.
    1A3,1/16;                          4A2,1/16
    1F3,1/4;                           4F2,1/4
    1A3,1/4;                           4F2,1/4
    1C4,1/4;                           4F2,1/4
    1F4,1/2;   2D4,1/2;   3A3,1/2;     4D2,1/2
    1A4,1/8.;  2E4,1/8.;  3A3,1/8.;    4C#2,1/8.
    1G4,1/16;             3G3,1/16;    4C#2,1/16
    1F4,1/4;              3F3,1/4;     4D2,1/4
    1A3,1/4;              3F3,1/4;     4D2,1/4
    1B~3,1/4;  2G3,1/4;   3E3,1/4;     4G2,1/4
    1C4,1/2;   2G3,1/2;   3E3,1/2;     4C3,1/2
    1C4,1/8;                           4C3,1/8
    1C4,1/8;                           4C3,1/8
 C  1A4,1/4.;  2F4,1/4.;  3C4,1/4.;    4F2,1/4.
    1G4,1/8;              3G3,1/8;     4G2,1/8
    1F4,1/4;              3F3,1/4;     4A2,1/4
    1E4,1/2;   2C4,1/2;   3G3,1/2;     4C3,1/2
    1D4,1/8.;  2B3,1/8.;               4B2,1/8.
    1E4,1/16;  2C4,1/16;  3G3,1/16;    4B2,1/16
    1F4,1/4;   2C4,1/4;   3A3,1/4;     4A2,1/4
    1F4,1/4;                           4F3,1/4
```

```
    1C4,1/4;                              4C3,1/4
    1A3,1/4;                              4A2,1/4
 DE 1F3,1/4;                              4F2,1/4
ENDSEG
SEGMENT 20
    1A4,1/8.M2;2F4,1/8.;  3A3,1/8.;   4F2,1/8.
    1A4,1/16M1;2F4,1/16;  3A3,1/16;   4F2,1/16
    1A4,1/4;   2F4,1/4;   3A3,1/4;    4F2,1/4
    1B4,1/4;   2F4,1/4;   3B3,1/4;    4G2,1/4
    1C5,1/4;   2F4,1/4;   3C4,1/4;    4A2,1/4
    1C5,1/2M2; 2F4,1/2;   3C4,1/2;    4A2,1/2
    1B4,1/8M1; 2F4,1/8;   3B3,1/8;    4G2,1/8
    1A4,1/8;   2F4,1/8;   3A3,1/8;    4F2,1/8
    1G4,1/4;   2E4,1/4;   3G3,1/4;    4C3,1/4
    1A4,1/4;   2F4,1/4;   3A3,1/4;    4F3,1/4
    1B4,1/4;   2E4,1/4;   3B3,1/4;    4G3,1/4
    1B4,1/2M2; 2G4,1/2;   3E4,1/2;    4C3,1/2
    1B4,1/4M1; 2E4,1/4;   3B3,1/4;    4C2,1/4
    1A4,1/4.M2;2C4,1/4.;  3A3,1/4.;   4F2,1/4.
    1G4,1/8M1; 2C4,1/8;   3G3,1/8;    4G2,1/8
    1F4,1/4;   2C4,1/4;   3F3,1/4;    4A2,1/4
    1E4,1/2M2; 2C4,1/2;   3G3,1/2;    4C3,1/2
    1D4,1/8M1;            3D3,1/8;    4B2,1/8
    1E4,1/8;   2C4,1/8;   3E3,1/8;    4B2,1/8
    1F4,1/4;   2C4,1/4;   3F3,1/4;    4A2,1/4
    1A3,1/4;   2F3,1/4;               4D3,1/4
    1B~3,1/4;  2G3,1/4;   3F3,1/4;    4F2,1/4
    1C4,1/2M2; 2G3,1/2;   3E3,1/2;    4C2,1/2
ENDSEG
* G
SEGMENT 30
    1C4,1/4M1;                        4C3,1/4
    1F4,1/4;   2C4,1/4;   3A3,1/4;    4F2,1/4
    1F4,1/4;   2B3,1/4;   3G3,1/4;    4G2,1/4
 H  1F4,1/8;   2C4,1/4;   3A3,1/4;    4A2,1/4
    1E4,1/8
    1D4,1/4;   2B3,1/4;   3F3,1/4;    4B2,1/4
    1D4,1/4;   2B3,1/4;   3F3,1/4;    4B2,1/4
    1D4,1/4;   2C4,1/4;   3F#3,1/4;   4A2,1/4
    1G4,1/4;   2B3,1/4;   3G3,1/4;    4G2,1/4
    1B4,1/8;   2B3,1/8;               4G2,1/8
    1A4,1/8;   2A3,1/8;               4A2,1/8
    1G4,1/8;   2G3,1/8;               4B2,1/8
    1F4,1/8;   2F3,1/8;               4B2,1/8
    1F4,1/4;   2C4,1/4;   3A3,1/4;    4C3,1/4
    1E4,1/4;   2C4,1/4;   3G3,1/4;    4C2,1/4
    1C4,1/8;                          4C3,1/8
    1C4,1/8;                          4B2,1/8
    1F4,1/4.;  2C4,1/4.;  3A3,1/4.;   4A2,1/4.
ENDSEG
SEGMENT 40
 I  1G4,1/8;               3G3,1/8;    4C3,1/8
    1A4,1/8;               3A3,1/8;    4F3,1/8
    1B4,1/8;               3B3,1/8;    4G3,1/8
ENDSEG
```

```
SEGMENT 50
 J  1C5,1/2;    2A4,1/2;    3F4,1/2;    4A3,1/2
    1F4,1/8;    2D4,1/8;    3A3,1/8;    4D3,1/8
    1G4,1/8;    2B~3,1/8;   3F3,1/8;    4D3,1/8
    1A4,1/4.;   2C4,1/4.;   3A3,1/4.;   4C3,1/4.
ENDSEG
SEGMENT 60
 K  1B4,1/8;    2F4,1/8;    3D4,1/8;    4C3,1/8
    +1G4,1/4;    2E4,1/4;    3B3,1/4;    4C3,1/4
ENDSEG
SEGMENT 70
    ++1F4,1/2;    2C4,1/2;    3A3,1/2;    4F2,1/2
ENDSEG
*
END
```

Statements in the Commands Section always begin in column 1, that is, starting at the left margin. Each of the commands recognized by INSNOTRAN is described in the sections below. Refer to Appendix A for a detailed description of the example score and how these available commands are typically arranged in a legitimate score.

Below is an index to the command descriptions. In the following pages, they have been arranged in logical rather than alphabetical order for ease of study.

SECTION	NAME	PURPOSE
2.4.5	ASSIGN	To assign instruments to specific voices and modes
2.4.1	CODEORG	To define the starting address of the object code
2.4.21	DRAWENV	To plot the overall amplitude envelope of an instrument
2.4.20	DRAWINS	To plot the waveforms used by an instrument
2.4.22	DRAWSET	To plot the waveforms in a waveform table set
2.4.12	ENDCMD	To indicate the end of the Commands Section
2.4.27	Hex input	To allow direct entry of hexadecimal object code
2.5.25	KEYBOARD	To automatically enter the "live keyboard" mode
2.4.23	LEFTCHAN	To assign specific voices to the left stereo channel
2.4.3	LIB	To specify the file name of an instrument library
2.4.4	LIBINS	To define an instrument from a library
2.4.16	MODIFY	To modify an instrument for a repeat percussion effect
2.4.26	MSG	To cause a message to be displayed when the score is played
2.4.13	NEWINS	To define a new instrument automatically
2.4.11	NVOICES	To specify the maximum number of voices that will sound
2.4.9	OCTAVE	To specify an octave offset for one or more voices
2.4.10	OFFSET	To specify a pitch offset for one or more voices
2.4.7	PAUSE	To wait until the Pause key is pressed before continuing
2.4.6	PLAY	To designate the segment in the Notes Section to be played
2.4.19	QSEQTAB	To define an arbitrary waveform sequence table in hex
2.4.18	QWAVESET	To build a waveform set without amplitude optimization
2.4.15	SEQTAB	To create a waveform sequence table
2.4.8	SKIP	To leave space in the Commands Section for patching
2.4.2	TEMPO	To define the tempo at which the music is to be played
2.4.24	TEMPRMNT	To specify an alternate tuning system
2.4.17	WARBLE	To define a sustained instrument with a warbling sound
2.4.14	WAVESET	To build a waveform table set

2.4.1 CODEORG

PURPOSE: To define the starting address of the object code.

SYNTAX: *CODEORG <start addr.>*

ARGUMENTS: *<start addr.>* = hexadecimal starting address of object code command string

DISCUSSION:

This statement allows the load address of the music object file created by INSNOTRAN to be altered from its normal value of $2000. This feature is used to allow INSNOTRAN to compile scores that will be played on other computers using the INSMUS interpreter. Note that the object code itself is not changed by changing the load address since all addresses internal to the object code are relative to the load address.

EXAMPLES:

CODEORG 0E00

Defines the origin of the object code to be $0E00. The file thus generated would be directly usable on an AIM-65 CODOS system with INSMUS.

NOTES:
1. This command may only be issued as the first command of an INSNOTRAN score. It is ignored if found anywhere else. If the first command is not a CODEORG statement, a default value for the object code origin, which is correct for use with INSPLAY on an MTU-130/140, is assumed.

2.4.2 TEMPO

PURPOSE: To specify the tempo (speed) at which the music is to be played.

SYNTAX: *TEMPO <num>/<den>=<duration>*

ARGUMENTS: *<num>/<den>* = any reasonable fraction of a measure such as 1/4
<duration> = number of milliseconds (thousandths of a second) corresponding to the designated fraction of a measure

DISCUSSION:

This statement is used to specify how quickly the following music should be played. Tempo is specified by relating a musical note duration (half-note, quarter-note, etc.) to the desired physical note duration in thousandths of a second. The tempo may be changed as often as desired by interspersing TEMPO statements with PLAY statements in the Commands Section.

EXAMPLES:

TEMPO 1/4=500

Specifies that a quarter note will last for 500 milliseconds (.5 second). This corresponds to 120 beats per minute if the quarter note is one beat.

TEMPO 3/8=700

Specifies that three eighth notes together will take 700 milliseconds. Thus a single eighth note will take 700/3 or 233 milliseconds.

NOTES:
1. The specified tempo applies to all segments played until another TEMPO statement is seen. Also, the tempo may be changed only between segments.

2. The slowest tempo available is 1/1=7600 or equivalent. Tempos faster than 1/1=500 or equivalent will become increasingly inaccurate.

3. The actual internal Tempo parameter is computed as:

T=(100*<den>*<duration>)/(25.5*<num>*S)

where S is the sample period in microseconds. One tempo period, then, equals the sample period (118, in this case) times the tempo parameter computed above with the result being in microseconds. Internally, all note durations, repeat periods, envelope segment durations, etc., are an integral

number of Tempo periods long. For example, the statement TEMPO 1/4=500 will result in a T value of 66 and thus a tempo period value of 7.788 milliseconds. All physical note durations and other internal timings then will be an integral number of 7.788 millisecond tempo periods long.

4. Speeding up the tempo not only sounds the notes more rapidly, it also makes the envelopes of the notes shorter and conversely for slowing the tempo down. Large differences between the tempo at which an instrument is defined and the tempo at which it is played may adversely affect the instrument's timbre.

2.4.3 LIB

PURPOSE: To specify the filename of the instrument library.

SYNTAX: *LIB <filename>*

ARGUMENTS: *<filename>* = name of library file to be used

DISCUSSION:

This statement identifies the name of an instrument library file if one is to be used. It must appear before any LIBINS statements are used. The library name may be changed as often as desired by inserting additional LIB statements. A LIB statement is not needed if all of the instrument definitions are inline as part of the score.

EXAMPLES:

```
LIB INSTFILE.I
```

Defines a file named INSTFILE.I as an instrument library file. Any LIBINS statement executed after this (until another LIB statement) will search INSTFILE.I for the desired instrument.

NOTES:
1. The file name specified in a LIB statement is subject to all the CODOS rules governing file names and defaults. The file will be assigned to Channel 3. The format of library files is discussed in Section 4.4.

2.4.4 LIBINS

PURPOSE: To define an instrument from a library.

SYNTAX: *LIBINS <insno> W<setno> <insname>*

ARGUMENTS: *<insno>* = ID number to be associated with the instrument
<setno> = ID number to be associated with the waveform table set that is built
<insname> = name of the instrument, as it is found in the instrument library file.

DISCUSSION:

This statement is used to define an instrument according to a named instrument definition in the instrument library specified by the last LIB statement. The *<insno>* argument is the ID number for the instrument that will be used in a subsequent ASSIGN statement which assigns the instrument to a

voice. It must be between 1 and 15 inclusive. The *<setno>* argument is needed for internal housekeeping and it too must be between 1 and 15. When using library entries exclusively for instrument definitions, it is generally acceptable to make *<setno>* equal to *<insno>*.

EXAMPLES:

```
LIBINS 12 W12 FLOOGLE
```

Instrument 12 is defined as the instrument named FLOOGLE as found in the current library file designated by the latest LIB statement. The waveforms used in defining instrument 12 are designated as waveform set #12.

NOTES:

1. This instruction actually inserts the INSNOTRAN statements needed to define the instrument, as found in the library file. A LIB statement must be executed prior to this instruction, or an error will result and the instrument definition will be ignored. If the requested instrument is not found in the library file, an error will be flagged and the statement ignored.

2. Instrument 1 is predefined by INSPLAY as a pure sine wave. It may, however, be redefined by a LIBINS or NEWINS statement. Instrument 0 is predefined as the "silent instrument" and may not be redefined.

2.4.5 ASSIGN

PURPOSE: To assign instruments to specific voices and modes.

SYNTAX: *ASSIGN [M<mode>] V<voiceno>=<insno>[; V...]*

ARGUMENTS: *<mode>* = Playing mode for which this assignment is valid (default=1)
<voiceno> = Voice to be assigned.
<insno> = Instrument ID number to be played by the voice.

DISCUSSION:

The INSNOTRAN system has four independent "voices", each of which can play in any one of three "modes". The ASSIGN statement associates a particular voice and playing mode with a particular instrument. After the assignment is made, whenever the specified voice plays a note in the specified mode, then the specified instrument sound is used to play that note. The assignment persists until changed by a subsequent ASSIGN statement. By default, all voices and modes are initially assigned to instrument #1 which in turn is defined by default as a sine wave. ASSIGN statements must be coded in the score after the instruments they refer to have been defined.

EXAMPLES:

```
ASSIGN V1=2; V2=2; V3=7; V4=8
```

Assigns instrument number 2 to voices 1 and 2, instrument 7 to voice 3, and instrument 8 to voice 4. These instruments must be defined before any notes are actually played with them. Mode 1 is chosen by default.

ASSIGN M2 V1=4; V2=4

Assigns instrument 4 to voices 1 and 2, mode 2. This means that whenever one of these voices is being played in mode 2, instrument 4 will sound. The assignment for voices 3 and 4 is unaffected.

NOTES:
1. 15 is the maximum allowable instrument ID number and 3 is the maximum allowable mode number. Zero is not allowed for either.

2. Only one mode may be specified in a single ASSIGN statement. If it is desired to make assignments for all three modes, then 3 ASSIGN statements will need to be used. For a description of playing modes, see section 2.5.

2.4.6 PLAY

PURPOSE: To designate the score segment in the Notes Section to be played next.

SYNTAX: *PLAY <segmentid>*

ARGUMENTS: *<segmentid>* = a number matching the ID of the segment to play next.

DISCUSSION:

The PLAY statement is the only statement that actually makes sound. It instructs the INSPLAY program to play the specified song segment coded in the Notes Section. After the segment is played, the next statement in the Commands Section is executed. Segments may thus be played as many times as desired and in any order simply by coding an appropriate sequence of PLAY statements.

EXAMPLES:

PLAY 10

Will play the group of notes given an ID of 10 in the Notes Section and then return to the Commands Section for the next command.

NOTES:
1. The segment ID given must match the ID of one of the segments in the Notes Section. If it does not, an error will be given at the end of the listing.

2. All of the notes between the corresponding SEGMENT and ENDSEG statements will be played with the current tempo, instrument assignments, and number of voices.

2.4.7 PAUSE

PURPOSE: To wait until the PAUSE key is pressed before continuing.

SYNTAX: *PAUSE*

ARGUMENTS: None.

DISCUSSION:

The PAUSE statement is used to halt the INSPLAY program when it is encountered until the PAUSE function key is pressed. It is useful for demonstrating the INSMUS-8 system or for separating song passages while debugging the score.

EXAMPLES: *PAUSE*

Halts INSPLAY until the PAUSE key is pressed.

2.4.8 SKIP

PURPOSE: To leave space in the Commands Section object code for patching.

SYNTAX: *SKIP <n>*

ARGUMENTS: *<n>* = number of bytes of memory to skip over

DISCUSSION:

This statement is used to simply skip over a specified number of bytes in the Commands Section object code. It is useful for leaving space that can be filled in later directly in the object code without having to recompile the song.

EXAMPLES:

```
SKIP 3
```

Go to the next statement and leave three bytes for user modification.

NOTES:
1. This command is intended for use by the experienced INSMUS-8 user who is familiar with the memory organization of the system. It outputs the specified number of "no-op" bytes ($EA). You can change these bytes after the song has been compiled to any codes recognized by INSPLAY desired.

2.4.9 OCTAVE

PURPOSE: To specify an octave offset for one or more voices.

SYNTAX: *OCTAVE V<voice>=<offset>[; V<voice>=...]*

ARGUMENTS: *<voice>* = voice number to be affected by the offset
<offset> = absolute octave offset from written value, -1 to +2

DISCUSSION:

The OCTAVE statement is used to automatically transpose the notes played by a specified voice or voices up (+) or down (-) a specified number of octaves. It is typically used to allow playing the entire 8 octave pitch range from C0 (about 15Hz) to C8 (about 4.1KHz) even though only a 5 octave range from C1 to C6 may be specified in note statements. Note that an OCTAVE offset is obsolute and will nullify any previous OCTAVE or OFFSET commands for the indicated voice.

EXAMPLES:

```
OCTAVE V3=1
```

Voice 3 is played one octave higher than written.

```
OCTAVE V4=-1; V2=+2
```

Voices 2 is played two octaves higher than written; Voice 4 is played one octave lower than written.

```
OCTAVE V1=0; V2=0; V3=0; V4=0
```

All voices played as written (nullifies all previous OCTAVE statements).

NOTES:
1. Voices not specified in an OCTAVE statement remain unchanged; any previous OCTAVE statements issued for them remain in effect.

2. An OCTAVE statement acts like an absolute offset -- that is, it nullifies all previous pitch and octave offsets.

2.4.10 OFFSET

PURPOSE: To specify a pitch offset for one or more voices.

SYNTAX: *OFFSET V<voice>=[R]<offset>[; V<voice>...]*

ARGUMENTS: *<voice>* = voice to be affected.
<R> = keyletter that indiciates a relative offset.
<offset> = pitch offset, in half-steps.

DISCUSSION:

The OFFSET statement is used to automatically transpose the notes played by a specified voice or voices up (+) or down (-) a specified number of semitones. It is typically used to allow direct coding of note statements from music written for a "transposing instrument" such as a B-flat trumpet. In such a case, the correct offset would be -2 for the voice assigned to the trumpet which would make the voice sound the pitch B-flat when it was instructed to play a C.

EXAMPLES:

```
OFFSET V1=2
```

Voice 1 is played one whole step above what is written.

```
OFFSET V4=-2 V2=R+2 V3=-8 V4=0
```

Voice 1 is played one whole step below what is written; Voice 2 is raised one whole step above its current pitch; Voice 3 is played 4 whole steps below what is written; Voice 4 is played exactly as written.

NOTES:

1. Relative offsets are cumulative -- that is, they add to any previous offsets, be they relative or absolute or OCTAVE offsets. The TOTAL offset must be between -12 and +24 half steps from the written pitch. If an OFFSET statement attempts to set an offset outside these bounds, an error message will be printed and the offending offset ignored. Absolute offsets, on the other hand, nullify any previous offsets, including OCTAVE offsets.

2. Any unspecified voices are left alone.

2.4.11 NVOICES

PURPOSE: To specify the maximum number of simultaneous voices that will sound.

SYNTAX: *NVOICES <#ofvoices>*

ARGUMENTS: *<#ofvoices>* = the maximum number of voices that will be sounding in the succeeding music. Minimum value is 1, maximum is 4.

DISCUSSION:

The NVOICES statement is used to alter the number of voices the music player expects to see coded in the compiled Notes Section statements. When the current number of voices is less than 4, only voice numbers from 1 to the number of voices inclusive may play notes. Reducing the number of voices merely assigns the unused voices to the "silent instrument" internally; the sample rate of the digital synthesis routine is not increased.

EXAMPLES:

NVOICES 4

Specifies that up to 4 voices may be played at once in the music to be played following this statement up to another NVOICES statement.

NVOICES 2

Specifies that only 2 voices will be playing in the following music.

NOTES:

1. Four voices are specified by default when INSPLAY is loaded; thus only a change from 4 requires an NVOICES statement.

2. Specifying fewer than 4 voices is typically used only to conserve memory in the compiled score since each musical event requires NVOICES+1 bytes of memory to encode. Unless your score becomes very large, just use the default NVOICES 4.

3. The NVOICES statement in the Commands Section must agree with the MAXVOICE statement in the Notes Section in effect when the notes are compiled.

2.4.12 ENDCMD

PURPOSE: To indicate the end of the Commands Section.

SYNTAX: *ENDCMD*

ARGUMENTS: None.

DISCUSSION:

This statement is used to separate the Commands Section from the Notes Section of the score. Misplacing it or leaving it out will likely result in a multitude of error messages.

EXAMPLES: *ENDCMD*

Defines the end of the Commands Section. The next statement will begin the Notes Section.

2.4.13 NEWINS

PURPOSE: To define a new instrument automatically.

SYNTAX:

NEWINS <id> W<set> N=<#wav> [A=<%>] <A>/<B>=<D> H<hrm#> <t>,<a>[;<t>,...][;H...]

ARGUMENTS: *<id>* = ID number to be associated with new instrument
<set> = ID number associated with waveform table set to be built
<#waves> = no. of waveform tables to be used
<%> = overall amplitude scaling factor, in percent, 0-100
<A>/<B>=<D> = average tempo, in same form as a TEMPO statement
<hrm#> = desired harmonic number between 1 and 127 inclusive, or noise component between 145 and 255 inclusive (see Note 3)
<t> = time coordinate, in millisecopnds
<a> = relative harmonic amplitude between 0 and 100

DISCUSSION:

The NEWINS statement is the most convenient way to define a new instrument from scratch. Instruments in the INSNOTRAN system are modelled as a family of amplitude envelope curves, one for each significant harmonic in the tone. For specification purposes, each of these curves is approximated by a series of straight lines. The endpoints of these line segments are defined by the *<t>* and *<a>* arguments and can be different for each harmonic. For implementation purposes, these line segments are further approximated as "staircases" where the total number of steps is equal to the *<#waves>* argument. The time positions of the "risers" for these steps must be the same for all harmonics but need not be equally spaced. The INSNOTRAN compiler will iteratively adjust the position and width of the steps to provide the closest approximation possible using the number of waveforms specified. A typical instrument definition will use 20-30 waveforms; there is room in memory for approximately 100 waveforms total assuming a moderate length score of 2-3 minutes.

EXAMPLES:

```
NEWINS 2 W12 N=8 A=75 1/4=500  H1 0,75; 2000,0;  H2 0,85; 500,30; 1500,0;
                               H3 150,0; 500,50; 2000,0
```

Defines Instrument 2 using 8 waveform tables which will be designated Waveform Set #12. The final overall amplitude is 75% of the optimized amplitude. The tempo for which this definition is exact is a quarter note being equivalent to 500 milliseconds. Harmonic 1 (called the fundamental) starts at 75% amplitude and decays to zero in 2 seconds. The second harmonic starts at 85% amplitude, decays rapidly to 30% after 500 milliseconds, and then decays more slowly to zero in another 1000ms (1500ms total). The third harmonic is absent for the first 150 milliseconds of the note. After 150 milliseconds it starts increasing to 50% amplitude until the 500 millisecond point. After 500ms it then begins to decay to zero over the next 1500ms. This example also illustrates the continuation of the NEWINS statement on the following line -- the only requirements are that the unfinished line must end in a semicolon, and the following line must not begin in column 1. Note how spaces have been added between harmonics and indentation used to improve readibility. Many more examples may be found in section 4.2.

NOTES:

1. The NEWINS statement relieves the user of the responsibility of building a waveform table set and then a sequence table to implement it as required in earlier versions of INSMUS. Harmonic components are entered in line segment form with a true time scale, which facilitates easy use of published data. The compiler then optimizes waveform table and sequence table usage to most closely match the desired curves. For the user who wants more control over the instrument definition process, the WAVESET and SEQTAB commands are available. See Chapter 4.

2. The Waveform Set Number is required for memory management purposes. This allows the user to overwrite an existing waveform set if memory needs to be conserved. Waveform Set Numbers can range from 1 to 16, inclusive.

3. Instrument 1 is predefined by the INSPLAY program as a pure sine wave. It may, however, be redefined by a NEWINS statement and will stay redefined until INSPLAY is reloaded. Instrument 0, the "silent instrument," may not be redefined, as it is used to silence unused voices.

4. The noise component is actually a squarish wave having a peak amplitude of 1/2 the value specified by <amp> and randomly determined "high" and "low" times influenced by the "harmonic" number. The resulting noise is less than ideal but is useful, for example, in simulating the "chiff" of organ pipes. An additional restriction on <harm#> besides the range mentioned above is that the number should not be a multiple of 16. See chapter 4 for additional information.

5. The average tempo parameter is required to associate time in milliseconds to waveform table counts. It in no way affects previous or subsequent tempo settings using the TEMPO command. Interesting effects, however, can be produced by playing an instrument much faster or slower than the tempo at which it was defined.

6. The maximum time value allowed is the number of milliseconds equivalent to a whole note (1/1) at the average tempo specified. It is important to be aware that most notes (anything less than a whole note, in fact) will not complete the entire envelope before being cut off. A trumpet-like instrument, for example, defined to end at the whole note value (2 seconds, for 1/4=500) will be cut off in the middle of its sustain if it were told to

play a quarter note. Likewise, if the same instrument were defined to end at the quarter note value (500 milliseconds in the example above), a half note would sound exactly like a quarter note followed by a quarter rest. A possible solution of this problem would be to have multiple definitions of such an instrument, each having a different sustain length. See Chapter 4.

7. The harmonic amplitude parameters merely give relative amplitudes among the harmonics. The compiler automatically optimizes the total amplitude to be the maximum value allowed with no overflow. This optimum amplitude is then scaled according to the A parameter, if given.

8. High-numbered harmonics should not be specified for instruments that will be played at high pitches, because of aliasing. See Chapter 4.

9. A Waveform Set can only be overwritten by one that is the same size or smaller. An attempt to redefine a Waveform Set using more tables than the original definition results in an error, and the definition is ignored.

2.4.14 WAVESET

PURPOSE: To build a waveform table set.

SYNTAX:

WAVESET W<setno> N=<#waves> [A=<%max>] H<harm#> <w>,<amp>[;<w>,<amp>...][;H...]

ARGUMENTS:
- *<setno>* = Waveform Set Number between 1 and 16 inclusive
- *<#waves>* = number of waveform tables to create
- *<%max>* = overall amplitude, in percent of optimized value (default=100)
- *<harm#>* = desired harmonic number between 1 and 127 inclusive or noise component between 145 and 255 inclusive (see Note 3)
- *<w>* = waveform number (X-coordinate of line segment) between 0 and <#waveforms>-1 inclusive
- *<amp>* = relative harmonic amplitude between 0 and 100% inclusive

DISCUSSION:

The WAVESET statement is used to automatically create a whole set of related waveforms automatically. Typically the harmonic content of each waveform in the set differs from its neighbor only slightly so that when the waveforms are played rapidly in sequence, a seemingly smooth variation in harmonic content is heard. A convenient way to model smooth harmonic content variations is to represent the amplitude envelope shape of each significant harmonic as a series of straight lines. The endpoints of these line segments are defined by the *<w>* and *<amp>* arguments and can be different for each harmonic. Note that the WAVESET statement only creates a waveform set, not a complete instrument like NEWINS does (see section 2.4.13). A SEQTAB statement that refers to waveforms in the set must follow the WAVESET statement to define an instrument using waveforms from the set. See section 4.3 for a detailed discussion of the WAVESET and SEQTAB statements.

EXAMPLES:

```
WAVESET W12 N=32 A=75  H1 0,100; 31,0;  H2 0,80; 29,0;  H3 0,64; 25,0;
                       H4 0,48; 21,0;   H5 0,32; 16,0;  H6 0,16; 8,0;
                       H7 0,0; 8,25; 31,0;    H8 3,0; 10,20; 31,10
```

Builds Waveform Set #12, containing 32 wavefroms. The final overall amplitude is 75% of the optimized value. Harmonic 1 (the fundamental) starts at maximum amplitude and decays to 0 in 32 tables. Harmonics 2-6 are all present at the beginning of the tone but fade out at different rates. The seventh harmonic starts at zero amplitude, increases to 25% during the first 8 tables, and then decreases back to zero during the remaining 24 tables. The eighth harmonic remains at zero until the third table at which point it increases toward 20% until the tenth table and then decreases to 10% through the 32nd table. This example also illustrates continuation of a WAVESET statement on the next line. The only restrictions are that the unfinished line must end with a semicolon, and the continuing line must not begin in column 1.

NOTES:

1. The Waveform Set ID is used for memory management. A waveform set may be overwritten by building another set with the same ID that is the same size or smaller than the original set. An attempt to overwrite with a bigger set causes an error, and the WAVESET command is ignored.

2. The harmonic amplitude parameters merely give relative amplitudes among the harmonics. The compiler automatically optimizes the total amplitude to be the maximum value allowed with no overflow. This optimum amplitude is then scaled according to the A parameter, if given.

3. The noise component is actually a squarish wave having a peak amplitude of 1/2 the value specified by <amp> and randomly determined "high" and "low" times influenced by the "harmonic" number. The resulting noise is less than ideal but is useful, for example, in simulating the "chiff" of organ pipes. An additional restriction on <harm#> besides the range mentioned above is that the number should not be a multiple of 16. See Chapter 4 for additional information.

4. High-numbered harmonics should not be specified for waveforms that will be played at high pitches, because of aliasing. See Chapter 4.

2.4.15 SEQTAB

PURPOSE: To define an instrument by creating a waveform sequence table.

SYNTAX: *SEQTAB <id> X=<x> BR=<br1> W<w1>:<sa>,<ea> [BR=<br2> W<w2>:<sr>,<er>]*

ARGUMENTS:
- *<id>* = ID number of instrument to be created.
- *<x>* = option/sustain parameter
- *<br1>* = block size/repeat parameter for attack (2-digit hex number, 1st digit = block size-1, 2nd digit = repeat-1)
- *<w1>* = waveform set reference number to be used for the attack
- *<sa>* = starting waveform number for attack in attack wave set
- *<ea>* = ending waveform number for attack in attack wave set
- *<br2>* = block size/repeat parameter for release (same format as br1)
- *<w2>* = waveform set reference number to be used for the release
- *<sr>* = starting waveform number for release in release wave set
- *<er>* = ending waveform number for release in release wave set

DISCUSSION:

Dynamically varying instrument sounds are created in the INSMUS-8 system by using sequences of waveform tables. When a note is sounded, each waveform in the sequence is played for a very short time and then the next waveform is

played and so on until the note is complete. The waveforms are nearly enough alike and the sequencing occurs rapidly enough so that the ear blends the individual waveforms together into a seemingly smooth variation much like the eye blends the frames of a film together.

A waveform sequence table is a table of pointers to waveforms. When a note is played, INSPLAY moves through the sequence table sequentially and uses the pointers to determine which waveforms are to be played when. In the INSMUS-8 system, a waveform sequence table and an instrument ID are synonymous. Waveform sequence tables are always 256 entries long and each entry corresponds to one tempo period of time which is 1/256 of a whole note duration. Usually far fewer than 256 different waveforms are actually used so many of the sequence table entries will point to the same waveform. The SEQTAB statement is used to automatically generate such sequence tables given a number of parameters. Please refer to section 4.3 for a detailed discussion of these controlling parameters.

EXAMPLES:

```
SEQTAB 14 X=5 BR=09 W3:1,32 BR=3A W4:17,1
```

Defines Instrument 14 by creating a waveform sequence table. The sustain is level and is 32*INT(5/2)=64 Tempo periods long. The attack sequence is taken from the first 32 waveforms of Waveform Set #3; B=1 and R=10. The release sequence is taken from the first 17 waveforms of Waveform Set #4, in reverse order; B=4 and R=11.

```
SEQTAB 8 X=0 BR=00 W1:4,15
```

Defines Instrument 8 by creating a waveform sequence table. Since X is zero, this is a simple stretch -- the release arguments are not needed. Twelve waveform tables are used, beginning with the 4th waveform in set #1. Block size = 1; repeat = 1.

NOTES:

1. This command is intended to be used by those experienced with the INSMUS-8 system. More information about building waveform sequence tables, especially concerning the X, B, and R parameters, is available in Chapter 4.

2. In most cases, the SEQTAB command will immediately follow a WAVESET command, but this is not necessary. If the Waveform Set Numbers specified for <w1> and <w2> have not been previously created in a WAVESET command, then unexpected or undesirable sounds can result. See section 4.3 for a thorough description of wavesets, sequence tables, and memory organization.

3. Instrument 1 is predefined by the INSPLAY program as a pure sine wave. It may, however, be redefined by a SEQTAB statement and will stay redefined until INSPLAY is reloaded. Instrument 0, the "silent instrument," may NOT be redefined, as it is used to silence unused voices.

2.4.16 MODIFY

PURPOSE: To modify an instrument for a repeat percussion effect.

SYNTAX: *MODIFY <dest id>=<source id> <period>*

ARGUMENTS: *<dest id>* = ID of instrument to be created
<source id> = ID of instrument used as source
<period> = repeat period, as a valid duration specification

DISCUSSION:

This statement is used to modify the waveform sequence table of an existing instrument to create a new sequence table, and thus a new instrument sound using the same waveform set as the old instrument. If the *<source id>* and *<dest id>* arguments are the same, then the new sequence table replaces the old sequence table. If they are different, then both sequence tables and thus both instruments are simultaneously defined and available for playing. The modification implemented by this statement simply takes the initial portion of the source sequence table and repeatedly copies it into the destination table thus giving a repeat percussion effect. The amount copied from the first table before repeating is determined by the *<period>* argument.

EXAMPLES:

```
MODIFY 4=3 1/16
```

Defines Instrument 4 based on Instrument 3, except with the first part of the envelope constantly repeated as sixteenth notes. If Instrument 3 is a "plucked string" type of instrument, this will create a strumming effect. Instrument 3 is unchanged.

```
MODIFY 7=7 1/32
```

Instrument 7 becomes a "strummed" version of itself, as repeated thirty-second notes. The previous definition of Instrument 7 is lost.

NOTES:
1. The source instrument must be previously defined.

2. Instrument 1 is predefined by the INSMUS program as a pure sine wave. It may, however, be redefined by a MODIFY statement and will stay redefined until INSMUS is reloaded. Instrument 0, the "silent instrument," may NOT be redefined, as it is used to silence unused voices.

3. See section 2.5 for information about duration specifications.

2.4.17 WARBLE

PURPOSE: To define a sustained instrument sound with a warbling sound.

SYNTAX: ***WARBLE*** *<dest>=<source> <fs> <bs>*

ARGUMENTS: *<source>* = Existing instrument ID number to be used as a model.
<dest> = ID number of warbling instrument to be created.
<fs> = Number of tempo periods to copy forward from source to destination in first half of each warble cycle.
<bs> = Number of tempo periods to copy backward from source to destination in second half of each warble cycle.

DISCUSSION:

This statement is used to modify the waveform sequence table associated with instrument number *<source>* and create a new waveform sequence table for instrument number *<destination>* which is a "warbled" version of the source. The modification works by first copying *<fs>* entries from the source table to the destination table and then copying *<bs>* entries backward from the current position in the source into the destination. This completes one "warble cycle" in the destination table. This is repeated for additional cycles starting at

the current position in each table until the destination table is filled. Typically *<fs>* and *<bs>* will be the same which means that the first part of the source instrument's envelope is repeated forward and backward in the destination instrument's definition. See Chapter 4 for additional information.

EXAMPLES:

WARBLE 5=4 10 10

The waveform sequence table for instrument number 5 is created from the sequence table for instrument 4 by taking the first 10 entries of the instrument 4 table and repeating them forward and backward in the instrument 5 table.

WARBLE 10=2 8 4

The waveform sequence table for instrument number 10 is created from the sequence table for instrument 2 by alternately moving forward 8 entries and backward 4 entries through the instrument 2 table until the instrument 10 table is full.

NOTES:

1. The source instrument table is not affected unless the destination equals the source. Using destination=source is permissable if *<fs>=<bs>*.

2. The period of the warble is *<fs>+<bs>* tempo periods. 256 tempo periods make up a whole note.

3. Instrument 1 is predefined by the INSPLAY program as a pure sine wave. It may, however, be redefined by a SEQTAB statement and will stay redefined until INSPLAY is reloaded. Instrument 0, the "silent instrument," may NOT be redefined, as it is used to silence unused voices.

2.4.18 QWAVESET

PURPOSE: To build a waveform set without amplitude optimization.

SYNTAX: *QWAVESET W<setno> N=<#waves> <hexadecimal number string>*

ARGUMENTS: *<setno>* = Waveform Set Number between 1 and 16 inclusive
<#waves> = Number of waveform tables to use
<hexadecimal number string> = String of 2 digit hexadecimal numbers which represent the compiled version of an equivalent WAVESET. The last number in the string must be 00.

DISCUSSION:

QWAVESET is typically used in an instrument library to define an instrument in object form so that it is very quickly compiled. The standard WAVESET and NEWINS statements may require compilation times of several seconds to a minute or more while their waveform tables are being amplitude optimized. QWAVESET provides a way to bypass this optimization for instruments whose definition is already known and not subject to change. See section 4.4 for more information on instrument libraries and how to prepare QWAVESET statements from compiled instrument definitions.

EXAMPLES:

```
QWAVESET W12 N=32  01 00 4D 1F 00 FF 02 00 3D 1D 00 FF 03 00 31 19 00 FF;
                   04 00 24 15 00 FF 05 00 18 10 00 FF 06 00 0C 08 00 FF;
                   07 00 00 08 13 1F 00 FF 08 03 00 0A 0F 1F 07 FF 00
```

This QWAVESET statement will produce the same waveform set as the example given in the description of WAVESET in section 2.4.14. Note that when continuation lines are needed that a semicolon should be placed at the end of the line to be continued.

2.4.19 QSEQTAB

PURPOSE: To define an arbitrary waveform sequence table in hex.

SYNTAX: *QSEQTAB <insno> W<setno> <hexadecimal number string>*

ARGUMENTS: *<insno>* = ID number of instrument to be created
<setno> = Waveform Set Number between 1 and 16 inclusive
<hexadecimal number string> = String of 2 digit hexadecimal numbers which represent the compiled version of the latter portion of a NEWINS statement. The last number in the string must be FF.

DISCUSSION:

QSEQTAB is typically used with QWAVESET in an instrument library to define a NEWINS instrument in object form so that it is very quickly compiled. The standard NEWINS statement may require compilation times of several seconds to a minute or more while its waveform tables are being amplitude optimized. QSEQTAB in conjunction with QWAVESET provides a way to bypass this optimization for instruments whose definition is already known and not subject to change. See section 4.4 for more information on instrument libraries and how to prepare QSEQTAB statements from compiled instrument definitions.

EXAMPLES:

```
QSEQTAB 5 W5 02 01 03 01 04 01 05 01 06 01 07 01 08 01 09 01 0A 02 0B 03 0C 02;
             0D 03 0E 03 0F 03 10 04 11 0D 12 0D 13 01 FF
```

This QSEQTAB statement will produce the same waveform sequence table as the example NEWINS statement that is analyzed in section 4.2.2. Note that when continuation lines are needed that a semicolon should be placed at the end of the line to be continued.

NOTES:

1. The QSEQTAB statement is actually compiled into an "arbitrary sequence table" instruction for the INSPLAY interpreter. See section 3.5.

2. The waveform table addresses given in the hexadecimal number string (first member of each number pair in the string) are relative to the first waveform table address referenced. The first waveform table address is assumed to be the first waveform in the specified wave set.

2.4.20 DRAWINS

PURPOSE: To automatically draw the waveforms used by an instrument.

SYNTAX: *DRAWINS <ins#>*

ARGUMENTS: *<ins#>* = ID number of instrument whose waveforms are to be drawn.

DISCUSSION:

This statement will cause the waveforms that the specified instrument uses to be plotted on the screen when this statement is executed by INSPLAY. This is useful during demonstrations of the system and while experimenting with instrument specifications.

EXAMPLES:

```
DRAWINS  3
```

Draws the waveforms used by instrument #3 when this command is encountered in the Commands section.

NOTES:
1. After the waveforms are drawn, INSPLAY lights the PAUSE function key legend and waits until the operator presses it before continuing.

2. See Chapter 4 for a discussion about waveforms.

2.4.21 DRAWENV

PURPOSE: To plot the overall amplitude envelope of an instrument.

SYNTAX: *DRAWENV <id>*

ARGUMENTS: *<id>* = ID number of the instrument to be drawn.

DISCUSSION:

This statement will cause the amplitudes of the waveforms that the specified instrument uses to be plotted on the screen when this statement is executed by INSPLAY. This is useful during demonstrations of the system and while experimenting with instrument specifications. The envelope shape drawn will appear as a stair-step approximation to a smooth envelope with the step size dependent on the number of waveform tables used in defining the instrument. The horizontal axis of the graph is time in tempo periods from 0 to 255 while the vertical axis is peak waveform amplitude from 0 to 100%. See section 2.4.2 for more information about tempo periods.

EXAMPLES:

```
DRAWENV 3
```

Plots the overall amplitude of instrument number 3.

NOTES:

1. The amplitude actually plotted is the peak amplitude in the waveform tables that define the instrument.

2. After the envelope is drawn, INSPLAY lights the PAUSE function key legend and waits until the operator presses it before continuing.

2.4.22 DRAWSET

PURPOSE: To plot the waveforms in a Waveform Table Set.

SYNTAX: *DRAWSET W<set#>[:<first>,<last>]*

ARGUMENTS: *<set#>* = ID number of the Waveform Set to be drawn.
<first> = First waveform number to plot.
<last> = Last waveform number to plot.

DISCUSSION:

This statement will cause the specified waveforms in the specified waveform set to be plotted on the screen when this statement is executed by INSPLAY. This is useful during demonstrations of the system and while experimenting with instrument specifications. The difference between DRAWSET and DRAWINS is that the former draws a waveform set (or a portion of one) while the latter draws those members of waveform set(s) that are being used by a particular instrument.

EXAMPLES:

```
DRAWSET W8:1,5
```

Plots waveform tables 1 through 5 of waveform set number 8.

```
DRAWSET W18
```

Plots all waveforms of waveform table set #18.

NOTES:

1. All waveforms are plotted on a single set of axes.

2. After the waveforms are drawn, INSPLAY lights the PAUSE function key legend and waits until the operator presses it before continuing.

3. See Chapter 4 for a discussion about waveform tables and waveform table sets.

2.4.23 LEFTCHAN

PURPOSE: To assign specific voices to the left stereo channel.

SYNTAX: *LEFTCHAN* *ALL* / *<nn>* / *NONE*

ARGUMENTS: *ALL* = all 4 voices assigned to left channel
<nn> = 2-digit number specifying which two voices to assign to left channel; other two voices assigned to right channel
NONE = all voices assigned to right channel

DISCUSSION:

This statement is used to control the assignment of voices to speakers in a stereo system. When using only the standard internal DAC in the MTU-130/140 computer, this statement will have no audible effect on the sound since both channels will be output through the internal DAC. Note that an even number of voices must be assigned to a channel. Thus in the INSNOTRAN system, a channel may have 0, 2, or 4 voices assigned to it but not 1 or 3.

EXAMPLES:

```
LEFTCHAN ALL
```

Assign all voices to left stereo channel.

```
LEFTCHAN 14
```

Voices 1 and 4 are assigned to the left stereo channel; Voices 2 and 3 are assigned to the right stereo channel.

```
LEFTCHAN NONE
```

Assign all voices are assigned to the right stereo channel.

NOTES:
1. In order for this command to have any effect, the host system must have two DAC's, one for each channel. See appendix H.

2. The default setting is '12'.

2.4.24 TEMPRMNT

PURPOSE: To specify an alternate tuning system for the following PLAY statements

SYNTAX: *TEMPRMNT #<no>*
or
TEMPRMNT N=<notes per octave> <frq 1>,<frq 2>,...,<frq N-1>,<frq N>

ARGUMENTS: *<no>* = Temperament number from the following list:
0 = Standard 12 tone equal temperament (default)
1 = Pythagorean tuning
2 = Just intonation
3 = Mean tone tuning
4 = Irregular (Wirckmeister's correct temperament #3)
<notes per octave> = Number of different pitches in the top octave for user specified tuning
<frq x> = Frequency parameter in decimal for a pitch in the top octave

DISCUSSION:

Temperament refers to the tuning of the note frequencies in a piece of music. Although equal temperament is nearly universal for contemporary music and to most ears is entirely adequate for older music, other temperaments are sometimes used. The TEMPRMNT (note the abbreviated spelling) statement allows the most common of these alternate tunings to be easily used. The alternate form also allows user specified tuning in which not only are the pitches variable, but the number of pitches in each octave is also variable. The effect of a TEMPRMNT statement persists for all following PLAY statements until another

TEMPRMNT statement is executed. See Chapter 5 for a complete discussion of temperament.

EXAMPLES:

```
TEMPRMNT #2
```

will specify that Just intonation should be used when executing succeeding PLAY statements.

```
TEMPRMNT N=19 7822, 7542, 7272, 7011, 6760, 6518, 6284, 6059, 5842, 5633,
              5431, 5237, 5049, 4868, 4694, 4526, 4326, 4207, 4056
```

will specify that a new tuning system based on the 19 tone equal tempered scale should be used when executing succeeding PLAY statements. Note that the statement is continued on the second line after a comma.

NOTES:
1. When the number of notes per octave is not 12, a note name translation table must be prepared to aid in entry of the correct note name for the pitch desired. See section 5.4 for an example.

2.4.25 KEYBOARD

PURPOSE: To automatically enter the "live keyboard" mode.

SYNTAX: *KEYBOARD*

ARGUMENTS: NONE

DISCUSSION: The KEYBOARD command is used to automatically stop playing and enter the live keyboard mode. This can be useful for live demonstrations of the INSMUS-8 system and for experimenting with instrument definitions. See section 3.3 for more information about the live keyboard mode of INSPLAY.

EXAMPLE:

```
KEYBOARD
```

will cause INSPLAY to stop playing music and enter pause mode with the live keyboard active. Playing may be resumed at the next statement by pressing the PAUSE function key.

2.4.26 MSG

PURPOSE: To cause a message to be printed on the screen by INSPLAY.

SYNTAX: *MSG '<string>'*
"<string>"

ARGUMENTS: *<string>* = Any string of ASCII text not including the delimiting ' or " whichever is used.

DISCUSSION:

This statement is used to display a message in the log area of the screen when INSPLAY encounters it in the Commands Section. It is useful for identifying sections of the score being played while it is being "debugged".

EXAMPLES:

```
MSG "Starting second movement."
```

will cause the message "Starting second movement" to be printed on the screen when INSPLAY executes this statement.

NOTES:
1. Usually enough time will be required to display the message that an audible break can be heard in the music.

2.4.27 DIRECT ENTRY OF HEXADECIMAL OBJECT CODE

If you want to enter INSPLAY commands directly in hexadecimal code, simply begin in column one with a dollar sign ($), followed by a string of two digit hexadecimal numbers (each between 00 and FF), each separated by one or more blanks and terminated with either a carriage return or a character that is not a hexadecimal digit. These numbers will be output directly to the object file, where the INSPLAY interpreter will act on them as commands. None of the automatic features offered by the INSNOTRAN compiler will be cognizant of the meaning of direct hex commands. However, direct coding might be useful for modifying or adding to existing song files which were prepared by hand coding in hexadecimal as required by earlier Instrument Synthesis systems. See section 3.5 for a description of the object code recognized by INSPLAY.

2.5 NOTES SECTION STATEMENTS

Statements in the Notes Section may either be Notes Section Commands or Note Statements. Commands must start at the left margin (column 1). If column 1 is blank or contains a digit or an event modifier, then the statement is assumed to be a note statement. Otherwise, it is compiled as a command statement. Don't confuse commands in the Notes Section with commands in the Commands Section, because they have different names and do different things.

2.5.1 END

PURPOSE: To end the Notes Section and thus the score.

SYNTAX: ***END***

ARGUMENTS: None.

DISCUSSION: The END statement must be the last statement in the score at the end of the Notes Section.

EXAMPLES: ***END***

Defines the end of the score and terminates the compilation. Must be present at the end of every score or an error message results.

2.5.2 ENDSEG

PURPOSE: To end a segment of note statements.

SYNTAX: ***ENDSEG***

ARGUMENTS: None.

EXAMPLES: ***ENDSEG***

Defines the end of the previous segment of music and will terminate the PLAY command that initiated playing of this segment.

NOTES:
1. Every segment of music in the Notes Section must be terminated by an ENDSEG statement. Even if the whole piece is coded as one segment, it must be terminated by an ENDSEG statement before being terminated by an END statement.

2.5.3 KEY

PURPOSE: To assign default pitch modifications for the following segment.

SYNTAX: ***KEY [<note>[#...][@...][; <note>...]]***

ARGUMENTS: ***<note>*** = letter name of note to be modified.
#... = one or more sharps
@... = one or more flats

DISCUSSION:

The KEY statement is used to relieve the user of the necessity of explicitly coding sharps and flats for every note that needs them. The way the key is specified is completely general and not restricted to the standard key signatures defined for the common major and minor 12 tone scales.

EXAMPLES:

```
KEY F#
```

Each time F is found in the following note statements, it is automatically changed to F#. This is usually called "the key of G" is musical jargon.

```
KEY B@; D##
```

B is flatted and D is double-sharped whenever they are encountered in the following note statements.

```
KEY
```

Disables automatic pitch modification. Each note sounds as it is written.

NOTES:
1. The statement remains in effect until another KEY statement is encountered. Unspecified pitches will have NO automatic modification, regardless of previous KEY statements.

2. Flats and sharps can be freely mixed. Any or all of the notes available (A-G) can be modified with a single KEY statement. If a note is named more than once, the last modification will be in effect.

3. Note statements entered in hexadecimal form are not affected by KEY statements.

2.5.4 MAXVOICE

PURPOSE: To specify the maximum voice number that will be used.

SYNTAX: ***MAXVOICE*** *<maxvoice#>*

ARGUMENTS: *<maxvoice#>* = the maximum voice number that will be used in the following note statements, between 1 and 4 inclusive.

EXAMPLES:

```
MAXVOICE 4
```

Specifies that voice numbers up to 4 will be used in the following music.

```
MAXVOICE 2
```

Specifies that only two voices will be coded in the following note statements.

NOTES:
1. The maximum voice number specified must match the NVOICES command in effect from the commands section when this segment is PLAYed.

2. The effect of a MAXVOICE command continues through succeeding segments until another MAXVOICE statement is seen.

3. MAXVOICE must only be used between segments, not within segments.

4. The default maximum voice number is 4.

2.5.5 RANGE

PURPOSE: To enable/disable automatic mode selection based on pitch.

SYNTAX: *RANGE V<voice#> [M<mode#>=<p1>-<p2>[; M...]]*

ARGUMENTS: *<voice#>* = number of voice which is to be affected, 1 to MAXVOICE
<mode#> = number of mode to take range assignment, 1 to 3
<p1>,<p2> = pitch specifications that denote the endpoints of the range to be assigned

DISCUSSION:

Each of the encoded notes in the object code produced by INSNOTRAN can specify one of 63 different pitches and one of three playing modes. Each of these modes may be assigned to a different instrument if desired thus allowing shifting around of instruments to occur on a note-to-note basis without having to end the segment and return to the Commands Section for each change.

This capability is typically used with instruments that should have different harmonic content in different pitch ranges (or "registers") to sound realistic. For example, low notes might use mode 1 which has been assigned to an instrument definition that is accurate for low notes, mid-range notes would use mode 2 which has been assigned to a different "mid range" instrument and similarly for high notes. The RANGE statement allows INSNOTRAN to automatically make the proper mode selection based on the note's pitch thus relieving the user of this responsibility. There is considerable flexibility in specifying how this is to be done as shown in the examples.

EXAMPLES:

```
RANGE V1 M1=B0-C2; M2=C#2-C4; M3=C#4-C#6
```

Implements automatic mode selection for Voice 1 over the entire range of the INSMUS program. If the note played by Voice 1 is between B0 and C2 inclusive, the instrument assigned to Voice 1, Mode 1 will be used. If the note is between C#2 and C4, the instrument assigned to Voice 1, Mode 2 will be played. If the note is between C#4 and C#6 inclusive, then Voice 1, Mode 3 will be selected.

```
RANGE V3 M2=G3-C5; M3=G4-C#6
```

Implements overlapping automatic mode selection for Voice 3 over a 2.5-octave range. If the note to be played by Voice 3 is between G3 and F#4 inclusive, then the instrument assigned to Voice 3, Mode 2 will be used. If the note is between C#5 and C#6 inclusive, then Voice 3, Mode 3 is selected. If the note is between G4 and C5 inclusive, then whichever automatic mode was last played is selected. If the note is outside these ranges, then the last mode specified by an 'M' modifier in the note string will be used.

RANGE V2 M1=G3-F#5

Implements automatic mode selection for Voice 2 over a 25-note range. If the note to be played by Voice 2 is within this range, Mode 1 will be selected. Otherwise, the mode depends on the last 'M' modifier used in the note string.

RANGE V4

Turns off the automatic mode select feature for Voice 4 only.

NOTES:
1. The voice specified by a RANGE statement must be assigned to an instrument for each mode indicated in the statement before any of the note statements following are PLAYed.

2. A RANGE statement nullifies any previous RANGE statement for the same voice.

3. The mode selection specified by the RANGE statement can be overridden by using the 'M' modifier in the note string. Automatic mode selection is resumed on the next event.

4. If the note to be played is outside the automatic selection range, and no 'M' modifiers have been specified, then the default M1 is used.

2.5.6 SEGMENT

PURPOSE: To identify the following segment of notes so that it can be PLAYed.

SYNTAX: *SEGMENT <segmentId>*

ARGUMENTS: *<segmentId>* = a number between 1 and 65535 used to uniquely identify this segment.

EXAMPLES:

SEGMENT 100

Identifies the following segment of music as segment number 100.

NOTES:
1. Any number between 1 and 65535 inclusive may be used for a segment ID. The value has no significance and need not be in ascending sequence. The only requirement is that no other segment have the same ID number.

2. Several SEGMENT statements may represent different "entry points" into the same group of notes and be terminated by a single ENDSEG statement.

When the first character of a statement in the Notes Section is not a letter or asterisk, the statement is assumed to be a note statement. When a note statement is so recognized, the compiler SKIPS columns 2-4, so they can contain whatever you choose. The compiler expects to find the first note or rest specification of the statement beginning in column 5.

A note specification consists of several digits, letters, and symbols all written together without any blanks between them. The first item in a note specification is the voice number that will actually play the note. In INSNOTRAN this must be a digit between 1 and MAXVOICE, which is usually 4.

Immediately after the voice number is the pitch specification. The musical pitch is specified by a keyletter which may be from A to G inclusive and must be upper case. Following the pitch letter may optionally be a pitch modifier -- a sharp (#), a flat (@), or a natural (~). The sharp simply raises the specified pitch by a half-step, while a flat lowers it by a half-step. Double-sharps and double-flats (whole step raise and lower) are also permitted simply by writing the # or @ twice. The natural nullifies any automatic pitch modification specified by a KEY statement. After the pitch letter or modifier is an octave number which indicates in which octave range the note should sound. Middle C (261.6 Hz) is represented by C4 and concert A (440 Hz) is represented as A4. Higher numbers represent higher octaves while lower numbers represent lower octaves. Octaves "turn over" at the C's, so the B just below middle C is B3. The lowest pitch available is B0 (30.87Hz) and the highest is C#6 (1108Hz). Octave and pitch offset statements in the Commands Section can be used to widen the overall pitch range down an octave to 15.43Hz and up two octaves to 4432Hz.

A comma separates the pitch specification just described from the duration specification. The note duration is represented by a fraction, such as 1/4 for a quarter note or 1/16 for a sixteenth note, etc. You may specify any reasonable fraction you like (numerator and denominator less than 256) that evaluates to unity (1/1) or less. INSNOTRAN doesn't check for weird fractions such as 2/5 or 3/11, so you can set up some strange rhythms or mimic imprecise durations if you like. Following the fraction you may optionally place a period which simply multiplies the previous fraction by 3/2 (a dotted note).

Following the duration specification comes an optional mode modifier. This consists of an upper case M, followed by a mode number between 1 and 3. This determines which instrument the specified voice will actually play. In order for a mode modifier to be valid, an ASSIGN statement must have been executed, assigning an instrument to the voice in question in the specified mode. If the automatic mode select feature is on, the mode modifier overrides it for this statement and this statement only. If the automatic mode select feature is not on, this mode carries through for this voice until another mode modifier is encountered for the same voice. This applies to subsequent note segments as well, so some care should be taken. The default mode value is 1; that is, if no mode modifiers are ever encountered, the note will use its Mode 1 assignment when not using the automatic select feature.

The following are some legal note specifications along with verbal descriptions. All are subject to pitch modification by a KEY statement, except those containing natural signs.

1C4,1/4	**Middle C, a quarter note played by Voice 1.**
3B@4,1/1	**B-flat above middle C, a whole note played by Voice 3.**
2F~#2,1/8.	**F-sharp 2 octaves below middle C (regardless of KEY), dotted eighth note, Voice 4.**

4C@5,1/4M2	C-flat above middle C (equiv. to B4), quarter note, Mode 2, Voice 4.
1F##4,1/4..M3	F-double-sharp (equiv. to G), double-dotted quarter note (equiv. to 1/4 x 3/2 x 3/2 = 9/16), Mode 3, Voice 1.
3G~5,1/24	G-natural an octave above middle C, a sixteenth note triplet (i.e., three of these take same time as two sixteenth notes), Voice 3.
2E5,1/5	E an octave above middle C, Voice 2, a fifth note.

The following specifications are illegal for the reasons noted:

5C4,1/4	Voice number is not 1, 2, 3, or 4.
3B4@,1/1M4	Sharp or flat must come before the octave number. Mode number is too big.
2F#2M1,1/1.	Mode modifier must come after duration specification. Final evaluated duration must not exceed whole note (1/1).
3H5,1/7	The pitch letter must be A-G.
1C4,1/72	Not really illegal, but the actual duration of such a short note may be substantially in error due to roundoff.

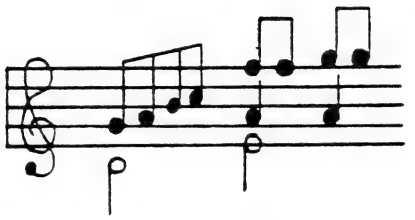
A rest specification consists of the keyletter R followed by a comma followed by the same kind of duration specification as is used for notes. In INSNOTRAN, rest specifications are needed only to specify complete silence (no notes playing), or to alter the "shortest duration" decision described below. Rests in individual voices occur automatically when no note is specified for the voice.

In INSNOTRAN, multi-part music (several voices playing at once) is entered a chord at a time. Consequently most note statements will consist of several note specifications on a line. When this occurs, you must put a semicolon (;) after each specification except the last one. This tells the compiler that another note specification follows and should be processed. Note that each note specification on the line must have a different voice number, since a voice may only play one note at a time in INSNOTRAN. The musical example below shows how a short series of chords would be coded:

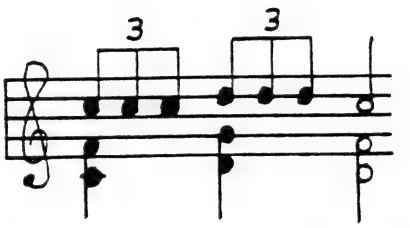
```
001 1C4,1/4; 2E4,1/4; 3G4,1/4; 4C5,1/4
    1C4,1/4; 2F4,1/4; 3A4,1/4; 4C5,1/4
    1C4,1/4; 2E4,1/4; 3G4,1/4; 4C5,1/4
    1B3,1/4; 2D4,1/4; 3G4,1/4; 4B4,1/4
    1C4,1/4; 2E4,1/4; 3G4,1/4; 4C5,1/4
```

When playing a note statement, all of the notes specified will start sounding their designated pitches at the same time. If all of the durations are the same, as in the above example, they will all stop sounding simultaneously as well. However, if one of the durations is shorter than the others, it will stop sounding first. When this happens, the computer will immediately move to the next line and start sounding the notes it specifies, even though some notes are still sounding from the previous line. If there are 3 or 4 different durations on a line, the computer continues after the shortest one has expired. If a rest specification is mixed in with note specifications on a line, the duration of the rest is included in the shortest duration search. If the rest is indeed the shortest, then the next line is played even though none of the notes on the previous line have finished.

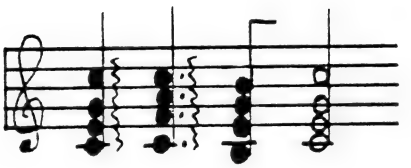
This one rule allows you to easily code things such as eighth notes against quarter notes, triplets, and other musical constructions. You must be careful not to start a new note with a voice before the note it was previously sounding expires; otherwise, you will get an error message and the previously sounding note will be cut short. Below are some musical examples and how they would be coded:



```
A:      1C4,1/2; 2G4,1/8
                 2A4,1/8
                 2B4,1/8
                 2C5,1/8
        1E4,1/2; 2A4,1/4; 3E5,1/8
                          3E5,1/8
                 2A4,1/4; 3F5,1/8
                          3F5,1/8
```




```
B:      1C4,1/4; 2F4,1/4; 3C5,1/12
                          3C5,1/12
                          3C5,1/12
                 1D4,1/4; 2G4,1/4; 3D5,1/12
                                   3D5,1/12
                                   3D5,1/12
                 1C4,1/2; 2F4,1/2; 3C5,1/2
```



```
C:      1C4,1/4;  R,1/24
        2E4,5/24; R,1/24
        3G4,4/24; R,1/24
        4C5,3/24
        1C4,9/25; R,1/24
        2F4,8/24; R,1/24
        3A4,7/24; R,1/24
        4C5,9/24
        1B3,1/8; 2E4,1/8; 3G4,1/8; 4B4,1/8
        1C4,1/2; 2E4,1/2; 3G4,1/2; 4C5,1/2
```

The compiler actually creates an object code string whenever there is a change in sound. A change defined as either the beginning or the ending of a note in at least one voice. Since changes in sound don't necessarily coincide with notes, we call the time between two changes an "event." The compiler, then, outputs one object code string for each event. The string consists of a duration byte and a number of sound bytes, one corresponding to each voice being used. Generally speaking, each INSNOTRAN note statemant corresponds to one event; the duration of that event is the duration of the shortest note or rest specification on the line.

"Event modifiers" allow easy manipulation of the duration of an event, without having to change the actual note specifications. There are two event modifiers. A plus sign (+) lengthens the event 12.5% while a minus sign (-) shortens the event by 12.5%. An unlimited number of event modifiers are allowed per line, and their effects are cumulative. Two plus signs would lengthen the event by 12.5% twice, resulting in a net lengthening of around 26.5% (not 25%!). The first modifier MUST be in column 5, and the string of modifiers is terminated by a blank or by a digit. Any +'s or -'s found after a blank or digit will be flagged as illegal characters. Be aware that the modifiers affect only the EVENT, not the entire line. Consider the following example:

```
111 +1C4,1/2; 2G4,1/4
```

In this case, the quarter note in Voice 2 would be lengthened by 12.5%, as well as the FIRST HALF of the half note in Voice 1. If there is no event modifier on the following line, then the second half of the half note will be played at normal speed. The entire half note, then, is only lengthened by 6.25%.

There may be cases in which the experienced user wishes to enter a note statement directly as object code. To do this, simply start with a dollar sign ($) in column 5. Then enter the coded string as hexadecimal numbers, separated by one or more blanks. Refer to section 3.5 to get information on note string object format. The compiler will expect to find exactly MAXVOICE+1 numbers before it encounters a carriage return or a character that is not a hexadecimal digit. If less then MAXVOICE+1 numbers are found, an error message is issued and the entire statement is ignored. If more numbers are found, the extra ones are ignored. Below is an example of hexadecimal coding:

```
$30 66 6A 6D 72  ;MAXVOICE is 4
$18 66 00 6D 72  ;Duration value first, followed by note values.
$18 26 6A 2D 32
++$60 66 6B 6F 72  ;Event modifiers are allowed.
```

3. THE INSPLAY MUSIC PLAYER

The INSPLAY program interprets music object code produced by the INSNOTRAN compiler and digitally synthesizes sound according to information in this code. To avoid delays (and audible breaks in the music) in getting the information it needs, the entire compiled "score" is read into memory ahead of time and then played from memory. The typical 3 minute score seldom exceeds 4K in size. Memory is also used for holding waveform tables which are also computed ahead of time for instant access. In fact, most of memory is typically used for waveform table storage. The following sections will thoroughly describe the INSPLAY program.

3.1 GENERAL OPERATION

INSPLAY is an interactive program that not only plays music under the preprogrammed control of a compiled score, but can also do numerous things under manual control. The music may be stopped, restarted, played one "event" at a time, played backward, "traced", terminated, and even edited in hexadecimal object form. Graphic displays of waveforms and envelopes of the instruments being used may be called up and also printed at any time. A "live" keyboard mode allows experimentation. Most of these things are be done while INSPLAY is in "PAUSE" mode. After such interaction is complete, the stored score may be restarted at the point where it was stopped.

Note that INSPLAY cannot play songs compiled with the SNOTRAN compiler that comes with all MTU-130/140 computers. It can, however, play INSMUS scores prepared for the the older INSMUS system on the KIM-1, SYM-1, AIM-65, PET, and MTU-130 computers. The only adjustment needed is that the INSMUS score should be loaded into memory starting at $2000.

3.1.1 Starting INSPLAY

INSPLAY is started by typing its name followed by an optional filename in response to a CODOS> prompt. If there is no filename following the command, then it is assumed that suitable music object code has already been loaded into memory starting at $2000. You may also have more than one filename in which case the second file will be loaded and played after the first is complete. For example, the following command:

```
INSNOTRAN SAMPLE.M
```

will cause INSNOTRAN to be loaded into memory and started. This in turn causes the music object file SAMPLE.M to be loaded into memory and the instructions it contains to be interpreted and carried out by INSPLAY. When the song is complete, INSPLAY will return to CODOS.

The command:

```
INSNOTRAN SAMPLE.M MODSAMPLE
```

will initially do exactly the same thing as the first example above. However, after the SAMPLE.M song is complete, the file called MODSAMPLE.C will be loaded into memory and it will be played. When it is finished, INSPLAY will look for additional filenames and upon finding none, will retrun to CODOS.

The command:

INSNOTRAN

Will load INSPLAY as before but now since there is no filename on the command line, INSPLAY will look directly in memory starting at $2000 for music object code to play. If music object code had not been loaded there previously, in all likelihood an error will be quickly encountered, an error message displayed, and a return to CODOS executed. If valid music object code is indeed found in memory, it will be interpreted just as if it had been read from a file. When the end is reached (or the ABORT function key used), INSPLAY will return to CODOS.

While INSPLAY is interpreting the score, some of the coded instructions are "logged" on the screen. These typically have to do with creating the instrument waveform tables in memory and assigning instruments to voices and do not normally appear while music is playing. See the figure in the next section for an example of what such a logout looks like.

When a WAVESET instruction (either from a WAVESET or a NEWINS statement) is being executed, the log output will display the relative memory page address (relative to the beginning of waveform memory) of the first and last waveform in the set. Then on the next line, the harmonic number it is currently working on will be displayed. Although it may seem to take awhile to compute the entire waveset, each time a harmonic number appears, the program has computed 256*N points on a sine wave and stored the results in tables where N is the difference between the waveform numbers. The construction of waveform sequence tables is also logged as "DEFINE INSTRUMENT #X". If the score has a MSG statement in it, the message is logged in a similar way.

3.1.2 Controlling the Playing Process

INSPLAY basically operates in two modes, Normal Mode and Pause Mode. In Normal Mode it is interpreting the meaning of the music object code and thus is either computing waveform tables, displaying waveforms, or actually making sound. The operator can influence the playing action somewhat while in normal mode. In Pause Mode, numerous options are available, some as separate function key "menus". In either mode, the 8 function keys on the keyboard and corresponding legends on the screen are used to make operator interaction easy.

In Normal Mode, the following functions are available:

Function key	Effect
ABORT	Active between each command and between each "event" of music. Results in a "USER ABORT, AT SONG ADDRESS XXXX, AT COMMAND ADDRESS YYYY" message where XXXX is the address in the Notes Section and YYY is the address in the Commands Section that was being interpreted when the ABORT key was pressed. INSPLAY then returns to CODOS or loads and begins interpreting the next song file.
BACKWRDS	Active only during playing of music "events" and causes events (notes) to be played backwards with normal tempo and instrument sounds as far back as the start of the current segment. It will not cross a segment boundary.

TRACE Displays the object code corresponding to each event before it is played as a 4 digit address in the Notes Section followed by N+1 hexadecimal bytes where N is the current value of MAXVOICE in the score. The first byte is the event's duration (20 = 1/8 note, 40 = 1/4 note, etc.) and the remaining bytes give the pitch and mode of each voice from 1 to N. 00 for a voice indicates a rest (silence). At the end of a segment, the address of the next PLAY statement in the Commands Section is displayed on a separate line and then the next segment is traced. Note that TRACE and BACKWRDS can be held down together.

PAUSE The program enters Pause Mode after the current command or music event is complete. The PAUSE legend appears in reverse field and the legends in the other function keys change to their Pause Mode functions. Pressing PAUSE again re-enters mormal mode and allows INSPLAY to continue. See below for details about Pause Mode.

KEYBOARD Enter the "live" keyboard mode after the current music event is complete. See section 3.3 for details.

The other program functions are executed while in Pause Mode. Below is a listing of the primary Pause Mode menu. Separate sections describe the edit mode and keyboard mode sub-menus.

Function key Effect

CUREVENT Play the current music event and also display it in the same format as Trace. This is done each time CUREVENT is released after being pressed.

BACKWRDS Play consecutive events backwards and also display them in the same format as Trace. The key may be held down for a continuous backwards display and play. If the beginning of the current segment is reached, the program "beeps" and will not cross the segment boundary.

FORWARDS Play consecutive events forwards and also display them in the same format as Trace. The key may be held down for continuous action. If the end of the current segment is reached, the program "beeps" and displays END SEGMENT, then waits for the key to be released. If FORWARDS is pressed again, the program crosses into the next segment but stays in PAUSE mode, thus allowing further tracing and editing.

PAUSE This legend is in reverse field, and when pressed, it causes the program to return to Normal Mode.

JMP STRT The program jumps to the start of the current segment and then plays and displays the first event. The user can therefore instantly go back to the beginning of the segment to review or re-edit.

JMP END The program jumps to the last event of the current segment, then plays and displays it. A warning "beep" sounds and the message END SEGMENT appears in reverse field. Pressing FORWARDS allows the system to cross into the next segment (if there is one).

EDIT The legends change to edit mode functions and EDIT is displayed in reverse field. See section 3.4.

INSPLAY is capable of displaying the waveforms and envelopes of the currently defined instruments. Such a display may be initiated either by instructions in the score resulting from a DRAWINS, DRAWSET, or DRAWENV statement, or interactively from the Keyboard menu.

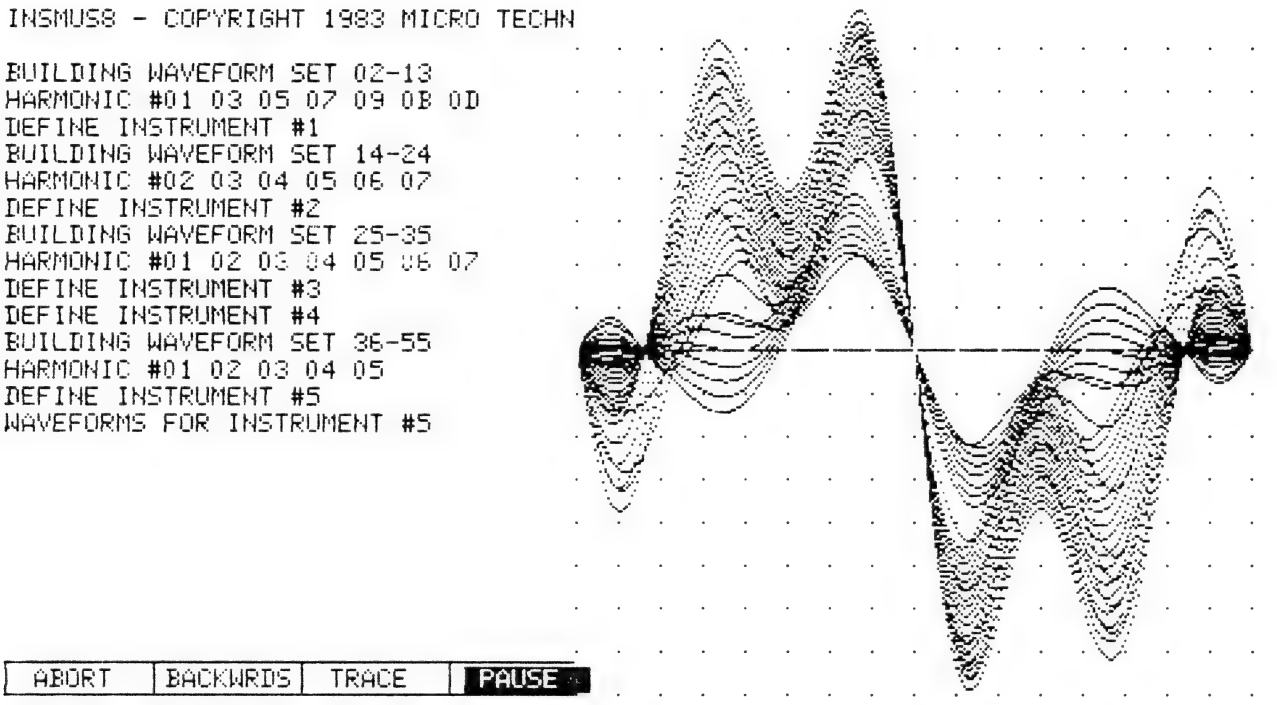
When a display results from execution of a DRAWINS, DRAWSET, or DRAWENV statement, the right half of the screen is cleared and the display is drawn there. After the plot is complete, the system enters Pause Mode. Normally one would want to continue with the score execution so the Pause function key (f4) would be pressed. Note that the display temporarily overlaps the legends for the second four function keys but they are still active. The display may also wipe out part of the system log output, most likely the log of a WAVESET instruction.

To force a waveform or envelope display when the score does not call for it, first press the KEYBOARD function key while INSPLAY is in normal mode. The live keyboard menu will be displayed (fully described in section 3.3). Instrument #1 is initially selected and a message to that effect is logged. To display the waveforms associated with the currently selected instrument, press the WAVEFRMS function key. All of the waveforms used by that instrument are displayed in sequence on the same set of axes. Only one cycle of each waveform is displayed. The display area is 256 points wide and 256 points high. The background grid has divisons 16 points wide and high. The horizontal axis represents time from 0 to 29.2 milliseconds and the vertical axis represents amplitude from -1.0 to +.999. To continue, press the f4 function key which has been relabelled PAUSE in reverse video. The keyboard mode menu will be restored when it is released.

To display the overall amplitude envelope of the currently selected instrument, press the ENVELOPE function key. The same display area is used but this time the horizontal axis is time from 0 to 255 tempo periods (see section 2.4.2 note 3) and the vertical axis is from 0 to 100%. The display typically shows as a "staircase" where each step represents the time spent on a single waveform table. The number of steps then is the number of waveform tables used although they are not necessarily unique.

To select a different instrument, enter its number as a single digit on the numeric keypad. Instrument numbers greater than 9 cannot be selected. A message is displayed whenever a new instrument is selected. If you try to select an instrument that has not been defined, a beep is sounded and instrument #1 is selected.

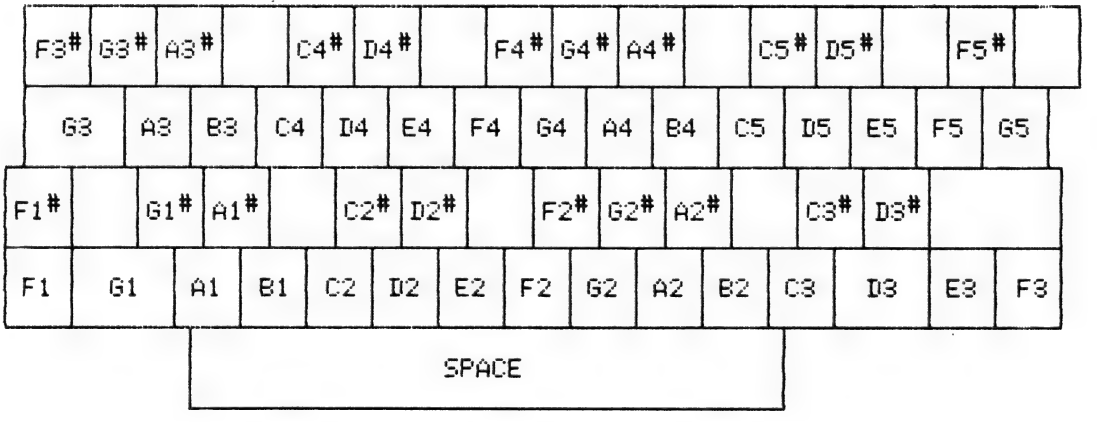
Pressing cntl/P while PAUSE is showing after a waveform or envelope display will cause a printout of the current screen contents. For this to function properly, a copy of the VMDUMP Utility for your printer must be resident on disk 0 and your printer must be connected and ready. VMDUMP is expected to load at $B400 (the standard utility area) and be less than $200 bytes long. The standard VMDUMP utilities from MTU meet these requirements. After the screen is printed, INSPLAY resumes what it was doing. Below is what such a screen dump would look like if cntl/P were pressed while the example score was playing:



3.3 THE KEYBOARD MODE

The keyboard mode can be entered either automatically by a command in the score (the KEYBOARD statement) or by pressing the KEYBOARD function key while INSPLAY is in normal mode. Upon entering keyboard mode, instrument #1 is automatically selected with zero octave and pitch offsets. The tempo (which influences the note envelopes) is set to its current value. The function key legends are also changed to show the available keyboard mode functions.

When in keyboard mode, the main keyboard becomes a pseudo music keyboard. Pressing a main keyboard key will play a note for as long as it is pressed. Due to hardware limitations, only one key at a time is recognized. Below is a diagram showing the correspondance between keyboard keys and musical notes (assuming that no octave or pitch offsets have been selected):



Below is a listing of the function keys in keyboard mode:

Function key	Effect
-8 va	The pitch offsetset is set to 0 (equivalent to OCTAVE -1)
NORMAL	The pitch offset is set to $0C (equivalent to OCTAVE 0). This is its initial value upon entry to keyboard mode.
+8 va	The pitch offset is set to $18 (equivalent to OCTAVE +1).
ENVELOPE	The amplitude envelope of the current instrument is displayed, then the PAUSE legend lights up in reverse field. During this pause the user may activate VMDUMP by pressing cntl/P. Pressing the PAUSE function key reverts the legends to the KEYBOARD mode legends (and erases the lowest portion of the display.
WAVEFRMS	The set of waveforms taken directly from the current instrument is displayed. After displaying the instrument waveforms, the system enters PAUSE mode as described for ENVELOPE above.
+1/2 va	The pitch offset is increased by one half-step (equivalent to OFFSET R+1). The highest offset available is $18 which is one octave up.
-1/2 va	The pitch offset is decreased by one half step (equivalent to OFFSET R-1). The lowest offset available is 0 which is one octave down.
RETURN	Leave keyboard mode and resume playing the score in normal mode.

Besides the function keys and the main keyboard, the numerical keypad keys may be used to select among the instruments currently defined. If an undefined instrument is selected, an error message is displayed and instrument #1 is selected. Also the cursor up and cursor down keys may be used to alter the tempo value. The new tempo value is displayed in hex. Holding a cursor key down will rapidly increment (cursor up) or decrement (cursor down) the displayed value. The fastest tempo is 02 and the slowest is FF. A typical tempo value is around 80.

It is also possible to save the keystrokes you enter in the live keyboard mode for later playback. This feature has not been integrated into the INSNOTRAN compiler but is explained in section 3.5.2.23.

3.4 EDITING THE SCORE IN MEMORY

One of the most useful features of INSPLAY when correcting a new score is its ability to edit the Notes Section object code directly in memory. The editing functions available include changing the duration and/or note pitches in the current event, inserting a new event, or deleting an extra event. INSPLAY will correctly "open up" or "close up" memory space in the score to accomplish the insert and and delete functions. Editing may only be performed on the compiled Notes Section code.

Edit Mode is entered from Pause Mode by pressing the EDIT function key. The Edit menu is then displayed with EDIT in reverse video. At this point, one of the first three function keys must be pressed to activate an edit function. You can leave Edit Mode without changing anything by pressing the EDIT key again which returns to the Pause menu.

The simplest edit function is Change. When the CHANGE function key is pressed, the address of the current event is displayed with blank contents and a flashing cursor appears at the first contents location. N+1 2 digit hexadecimal numbers should be entered for the desired contents at that address and then Return pressed to enter the change. The altered event is played and the Pause menu re-entered. The CODOS line editor is in control during the typein process so you can backspace, forward space, correct mistakes by overtyping, and even use cntl/B to recall a previously entered event.

When the DELETE function key is pressed, the current event is immediately deleted. The next event in sequence is then displayed and played. Note that since the current event has been deleted, the address shown for the next event is the same as the one shown previously for the current event. After the deletion and display, the Pause Mode menu is re-entered. If the last event in a segment is deleted, the display and play of the next event in sequence will be in error. The deletion, however, did occur correctly.

When the INSERT function key is pressed, the action is similar to CHANGE in that the address of the current event is displayed with no contents and with a flashing cursor. The data in the Notes Section starting with the current event and continuing to the end of the score will be moved forward in memory leaving room for the event about to be typed in. Thus the new event is inserted before the current event. When return is pressed, the new event entered will be stored, displayed, and played. Following this, Pause Mode is reentered.

One caution when using the Insert and Delete functions is that INSPLAY needs to know where the Notes Section ends in memory. This information is obtained from CODOS and will be correct only if the score was loaded into memory either by an explicit GET command immediately before playing or by having the score filename on the INSPLAY command line. If the score was loaded into memory by any other means (such as by a user program), or was loaded before INSPLAY was loaded, INSERT and DELETE have the potential to wipe out memory.

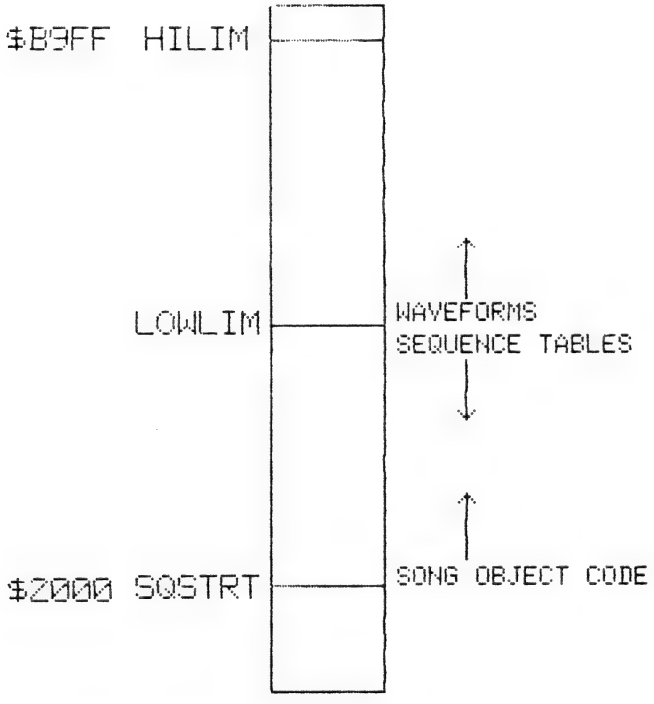
Also, there is a limit to how much can be inserted before the the score starts overflowing into the waveform area of memory. In general, you should look at the last object code address used in the INSNOTRAN listing of the song (the address to the left of the END statement) and restrict insertions such that they will fit into the remainder of that last memory page. If desired, you may put a number of lines with nothing but a Rest specification at the end of the Notes Section to reserve space in memory for insertions. If insertion overflow does occur, it will first wipe out the waveform sequence table of the highest numbered instrument in the score.

3.5 OBJECT CODE FORMAT

In many respects, INSPLAY emulates an "intelligent music synthesizer" chip which fetches "instructions" from memory and executes them one at a time. These instructions always have a single "op code" byte followed by address and data bytes. Most instructions have a fixed length but several have a variable length. The following gives a brief description of the object code format for those who wish to patch music object code in memory or "decompile" some of the songs on the INSMUS-8 Distribution Disk. There are a couple of instructions recognized by INSPLAY that have no INSNOTRAN equivalent statement. These may be experimented with by patching them into a compiled song by hand, perhaps in space skipped by a SKIP statement.

INSPLAY itself uses a fairly small amount of memory which is from $0700-$1FFF and $B900-$BBFF. The large gap between $2000 and $B8FF is thus free for compiled song data, waveform tables, and waveform sequence tables. Three variables are used internally in allocating this one block of memory among these 3 "users". SQSTRT is the address of the beginning of the block and thus its value is $2000 in MTU-130/140 INSPLAY. It is the address where the music object code is loaded and is also the address of the very first instruction in the Commands Section. Conversely, HILIM is the address of the top end of the memory block and thus its value is $B900 (one more than the end address). Nothing is ever stored at or above HILIM.

The third variable is LOWLIM and it is set somewhere between SQSTRT and HILIM. Waveform tables start at LOWLIM and grow upwards toward HILIM. Waveform sequence tables start immediately below LOWLIM and grow downwards towards SQSTRT. The score starts at SQSTRT and grows up toward the lowest waveform sequence table. This is illustrated below:



When INSNOTRAN has compiled a score, it knows how large the score is and how many waveform sequence tables (instruments) have been defined. It thus knows where LOWLIM should placed so that the space available for waveform tables is maximized while preventing the waveform sequence tables from ever overlapping the song object code. The instruction to set LOWLIM is always generated as the first instruction in the object code. The SETLOWLIM instruction is described below.

Many of the Commands Section instructions have an address field. Most addresses in the music object code are relative to SEQSTART which is also the load address of the object file. This allows the contents of the object file to be position independent and thus runnable on any of the many versions of INSMUS. You should also note that often, the contents of an address field will not be known when a statement is compiled and the object code is printed on the listing. In these cases the printed object code will have zeroes in the address field. At the end of the compilation when these addresses are known, INSNOTRAN actually "patches up" the object file on disk so that all of these deferred address fields have the proper contents. Thus when an address field in

the listing contains zero, you will have to load the object file and look in memory to determine what the real address is.

3.5.2 Commands Section Instructions

The following instructions are the result of compiling statements in the Commands Section. They are listed in op code sequence.

3.5.2.1 OUTPUT TEXT MESSAGE - E0 <text bytes> 00

This instruction displays the <text bytes> on the screen in the log area and then continues. Any ASCII codes are allowed in the text bytes including control characters. This instruction is generated by the MSG statement.

3.5.2.2 DISPLAY INSTRUMENT WAVEFORMS - E1 <ID>

This instruction causes all of the waveforms referred to by the waveform sequence table associated with instrument ID to be displayed on the same set of axes. ID should be between 00 and 0F inclusive. INSPLAY enters Pause Mode after the waveforms are displayed. This instruction is generated by the DRAWINS statement.

3.5.2.3 DISPLAY INSTRUMENT ENVELOPE - E2 <ID>

This instruction causes the overall amplitude envelope of the instrument ID to be displayed. ID should be between 00 and 0F inclusive. INSPLAY enters Pause Mode after the envelope is displayed. This instruction is generated by the DRAWENV statement.

3.5.2.4 DISPLAY WAVEFORM SET - E3 <SS> <EE>

This instruction causes all of the waveforms between SS and EE inclusive to be displayed on the same set of axes. The waveforms are designated by their address relative to LOWLIM. INSPLAY enters Pause Mode after the waveforms are displayed. This instruction is generated by the DRAWSET statement.

3.5.2.5 SKIP - EA (no operands)

Skip to the next byte. The SKIP instruction is useful for leaving space in the Commands Section. N SKIP instructions are generated by an INSNOTRAN "SKIP N" statement.

3.5.2.6 PAUSE - EB (no operands)

When INSPLAY encounters this instruction, it enters Pause mode and remains there until the operator presses the PAUSE function key. The PAUSE instruction is generated by the INSNOTRAN PAUSE statement.

3.5.2.7 JUMP - EC <HH> <LL>

This instruction will jump to location location HHLL relative to SQSTRT. Note that the address bytes are in natural order. The JUMP instruction might be used during patching or hand coding of object code. It could also be used to set up an "infinite loop" in which the same part of the score is repeated indefinitely. JUMP instructions are not generated by INSNOTRAN.

3.5.2.8 JUMP TO SUBROUTINE - ED <HH> <LL>

Execute a user supplied subroutine at location HHLL relative to SQSTRT. The SUBROUTINE instruction expects to find a 6502 machine language coded subroutine at that address. The subroutine may use any registers it wants but it must return with Y=0, decimal mode off, and interrupts disabled. This instruction is not generated by INSNOTRAN.

3.5.2.9 SET LOWLIM - EE <HH>

Set the value of LOWLIM (start of waveform memory) at the page address HH relative to SQSTRT. LOWLIM should be set such that the entire score object code plus all of the waveform sequence tables will fit into the space between SQSTRT and LOWLIM. SET LOWLIM is always the first instruction generated by INSNOTRAN.

3.5.2.10 SET HILIM - EF <HH>

Set the value of HILIM (end of waveform memory) at the page address HH relative to SQSTRT. Use of this instruction may make the score system dependent since not all implementations have the same amount of memory available. INSNOTRAN does not generate SET HILIM instructions.

3.5.2.11 SET PLAY ROUTINE AND NUMBER OF VOICES - F0 <SV>

This instruction establishes which internal "play routine" will be used to interpret the Notes Section code and how many voices will be coded in the Notes Section code. The S nybble (upper 4 bits) specifies one of 16 possible play routines. INSPLAY currently has only one play routine so S should always be zero. The V nybble (lower 4 bits) specifies how many voices the play routine should expect. This instruction is generated by the NVOICES statement.

3.5.2.12 ASSIGN - F1 <PM> [<IJ> <KL>] (1 or 3 byte operand)

This instruction assigns instruments to voices, modes, and stereo channels. Nybble P is the stereo pattern and nybble M is the mode for which the following instrument assignments apply. Nybbles I, J, K, and L, if present, are the instrument IDs for voices 1 through 4 respectively.

P sepcifies the stereo pattern according to the table below. Any value not listed is illegal and will cause an error exit.

P (hex)	P (binary)	Channel Assignments
0	0 0 0 0	Voices 1-4 all to left channel
3	0 0 1 1	Voices 1 & 2 to left channel, 3 & 4 to right
5	0 1 0 1	Voices 1 & 3 to left channel, 2 & 4 to right
6	0 1 1 0	Voices 1 & 4 to left channel, 2 & 3 to right
9	1 0 0 1	Voices 2 & 3 to left channel, 1 & 4 to right
A	1 0 1 0	Voices 2 & 4 to left channel, 1 & 3 to right
C	1 1 0 0	Voices 3 & 4 to left channel, 1 & 2 to right
F	1 1 1 1	Voices 1-4 all to right channel

Nybble M specifies the playing mode for which the following assignments apply. If M=0, then the next two bytes should not be included. If M=1, 2, or 3, then the next 2 bytes (4 nybbles) specify the instrument ID to be assigned to the 4 voices respectively for that mode. Three of these instructions (all specifying the same stereo pattern) are necessary to make instrument assignments in all 3 modes. An instrument ID of zero specifies the silent instrument.

This instruction with an M nybble of zero is generated by the LEFTCHAN statement. An ASSIGN statement will generate this instruction with appropriate values for the M, I, J, K, and L nybbles and a stereo pattern consistent with the last LEFTCHAN statement or P=3 if no LEFTCHAN statement has been processed.

3.5.2.13 ASSIGN TEMPO - F2 <TT>

This instruction establishes the duration of one tempo period. TT is the number of 114 microsecond sample times in one tempo period. Valid values of TT are $02 through $FF which gives a tempo period time of .228 to 29.07 milliseconds. This instruction is generated by the TEMPO statement.

3.5.2.14 SET OCTAVE OFFSET - F3 <OOOO> (one byte operand)

This instruction assigns an octave offset (-1 to +2) for voices 1-4. The one byte operand is a set of four nyps (pairs of bits) where each nyp refers to a voice (1 to 4 left to right) and the numerical value of the nyp (0, 1, 2, or 3) is the number of octaves plus 1 to offset that voice. The octave offset instruction cancels all previous octave and pitch offsets for all 4 voices. This instruction is generated for the OCTAVE statement.

3.5.2.15 SET PITCH OFFSET - F4 <P1> <P2> <P3> <P4> (4 byte operand)

This instruction applies half-step pitch offsets to voices 1-4 respectively. The offset can be absolute, relative positive, or relative negative. The pitch offset bytes are interpreted as follows:

$00 - $24 is absolute pitch offset in half-steps, $00=-1, $0C=0, $24=+24
$40 - $4F is relative pitch offset upward in half-steps, $40=0, $4F=+15
$80 - $8F is relative pitch offset downward in half-steps, $80=0, $8F=-15

This instruction is generated by the OFFSET statement.

3.5.2.16 CONSTRUCT WAVEFORM SET - F5 (multi-byte operand)

This instruction constructs a set of related waveforms according to straight line segment specifications of the amplitude envelopes of individual harmonics. The complete format of the instruction is shown below:

```
F5 <SS> <EE> <HN> <PG><AM> [<PG><AM>...] FF
                <HN> <PG><AM> [<PG><AM>...] FF
                  .
                  .
                  00
```

SS is the memory page number relative to LOWLIM that will receive the first waveform in the set. EE is the memory page number that will receive the last waveform in the set. The total number of waveforms in the set then is EE-SS+1. An error is signalled if EE is less than SS and if EE is such that waveforms would be created beyond the end of memory as defined by HILIM.

Following these bytes is coding for line segment approximations of the desired envelope shape for various harmonics of the tone. The harmonics need not be in any particular order except that "noise harmonics" must be coded after all normal harmonics. A harmonic specification starts with the HN byte and ends with a hex FF. HN specifies the harmonic number which may be from $01 to $7F (1 to 127 decimal). Values of $80 or more specify a noise component. Following HN are pairs of bytes, one for each breakpoint of the line segment approximation of the envelope shape desired for the harmonic. PG specifies the abscissa (X-coordinate) of the breakpoint as a memory page number relative to SS. AM specifies the ordinate (Y-coordinate) of the breakpoint which is the harmonic amplitude corresponding to that point in time. INSPLAY will then linearly interpolate the amplitudes at intermediate page addresses between the previous PG AM pair and the current one. In the sequence of PG AM pairs, PG must always increase and PG may not become greater than EE or an error is signalled. The end of a harmonic specification is signalled by coding an FF for PG and omitting the AM byte. At this point, INSPLAY expects to see another HN for the next harmonic. A 00 for HN signals the end of the instruction.

This instruction is generated by WAVESET, NEWINS, and QWAVESET statements. In the case of NEWINS, the Construct Waveform Set instruction is generated first followed by an Arbitrary Sequence Table instruction (see 3.5.2.19).

3.5.2.17 BUILD SEQUENCE TABLE - F6 <XI> <BR_1> <SA> <EA> [<BR_2> <SR> <ER>]

This instruction builds a waveform sequence table representing an instrument with an ID equal to the I nybble with attack stretch characteristics BR_1 using attack waveform set SA through EA and (optionally) release stretch characteristics BR_2 using release waveform set SR through ER. BR_2, SR, and ER are omitted if X=0. If X is odd, a level sustain is used, otherwise a warbled sustain is used. The sustain duration in either case is 32*INT(X/2). In the BR bytes, B is the initial block length and R is the number of times a block length is used before being incremented by 1. See section 4.3.2 for additional discussion about the B and R parameters. This instruction is generated by a SEQTAB statement.

3.5.2.18 MODIFY WAVEFORM SEQUENCE TABLE - F7 <DS> <NN>

This instruction modifies the waveform sequence table whose ID is S for a repeat percussion effect and constructs a new waveform sequence with an ID of D. The repeat period is NN. Basically the instruction uses the first NN entries from the S table and repeats them in the D table until the entire 256 entry destination table is completed. The S table is not changed. D may be equal to S in which case it replaces the S table. This instruction is generated by the MODIFY statement.

3.5.2.19 ARBITRARY SEQUENCE TABLE - F8 <ID> <WV> <NN> [<WV> <NN>...] FF

This instruction will construct an arbitrary waveform sequence table for instrument ID using waveforms WV repeated NN times each. Waveforms are identified by their page address relative to LOWLIM. The table is filled from the beginning with NN repetitions of the designated waveform. Additional WV NN pairs are coded for additional waveforms. An FF byte for WV signals the end of the instruction. Normally the sum of the NNs would be 256 but if it is less, The sequence table is filled to the end with repetitions of the last waveform. An error is signalled if the sum of the NNs is greater than 256. This instruction is created as the second part of a NEWINS statement and by a QSEQTAB statement.

3.5.2.20 CREATE WARBLED INSTRUMENT - F9 <DS> <FF> <BB>

This instruction creates a new instrument (waveform sequence table) D by copying alternately forward FF number of waveforms from the S waveform sequence table and backward BB number of waveforms. FF must be equal to or greater than BB. D may equal S if FF=BB. This instruction is generated by the WARBLE statement.

3.5.2.21 CHANGE TEMPERAMENT - FA F<I> or FA <NN> <HH> <LL> [<HH> <LL>...]

This instruction changes the internal note frequency table. If the high nybble of the first operand byte is F then the low nybble specifies the ID of a prestored temperament according to the list below:

0 = Standard 12 tone equal temperament
1 = Pythagorean tuning
2 = Just tuning, Ramis monochord
3 = Mean tone tuning, Aron's
4 = Baroque irregular temperament, Wirkmeister's #3

If the first byte is less than $62, then the next 2*NN bytes define the "top octave" of a microtonal scale in 2 byte pairs (double precision integers). There will be NN pitches in an octave. Refer to Chapter 5 for details. This instruction is generated by the TEMPRMNT statement.

3.5.2.22 ENTER LIVE KEYBOARD MODE - FC (no parameters)

This instruction causes an automatic entry into the live keyboard mode just as if the operator had pressed the KEYBOARD function key. When the operator presses the RETURN function key, interpretation resumes with the next instruction. This instruction is created by the KEYBOARD statement.

3.5.2.23 ENTER LIVE KEYBOARD MODE WITH SAVE - FD <HH> <LL>

This instruction is the same as FC above except that music key entries including durations are saved in memory at address HHLL relative to SQSTRT. No "rests" are saved, allowing the user to pause between each note. No check is made for the validity of HHLL. By following this instruction with an F0 01 (set number of voices = 1) and then FE HH LL, the user can hear (and edit) the data just keyed in live. INSNOTRAN has no statement to generate this instruction.

3.5.2.24 PLAY NOTE STRING SEGMENT - FE <HH> <LL>

This instruction plays the notes segment starting at address HHLL relative to SQSTRT. The notes are played event-by-event until a zero duration byte is encountered at which point Commands Section interpretation continues at the next instruction. This instruction is generated by the PLAY statement.

3.5.2.25 END - FF or 00 (no operands)

This instruction marks the end of the Commands Section. INSPLAY will return to CODOS or load and play the next object code file when it encounters this instruction. The END statement generates this instruction.

3.5.3 Notes Section Data

Object code generated by the Notes Section statements consists of events which are grouped into segments. An event is created by the INSNOTRAN compiler for each change in the chord structure of the musical sound implied by the score. There may be more than one event created for each line of note specifications in the score. In the Notes Section, only note statements and ENDSEG statements actually generate any object code. The other statements simply influence the object code generated by the note statements.

The first byte in each event is the duration byte. If the duration byte is zero, it is interpreted as the end of the segment and the pitch bytes are not needed. If it is non-zero, it specifies the duration of the event in tempo periods. INSNOTRAN performs scaling such that a duration byte of $FF specifies a whole note duration, $80 a half note, $40 a quarter note, etc.

The remaining bytes in each event specify the mode and the pitch to be played by each voice. The number of pitch bytes expected is equal to the number of voices specified by the last Set Number of Voices ($F0) instruction. Each pitch byte is divided into the high 2 bits and the low 6 bits. If the high two bits are non-zero, they specify that the voice should begin an attack using the instrument currently assigned to the voice and mode equal to the binary value of the two bits. If the high two bits are zero, then the voice should continue the envelope from the previous event. The low 6 bits specify the pitch to be played by the voice. With no octave or pitch offsets in effect, the pitch range specifiable is B0 ($01) up through C6# ($3F). By use of octave and pitch offsets, this range is extended 1 octave downward and two octaves upward overall.

4. INSTRUMENT CODING PRINCIPLES

Instrument definitions in INSNOTRAN encompass all the features of the older Instrument Synthesis Software Package (INSMUS). The very flexibility of the sound generation process used in INSMUS can make the task of defining a desired instrument seem rather complex. This is particularly true if the user is not familiar with the physical principles involved and/or has not had experience with sophisticated sound synthesizers before. For users who wish to experiment with constructing their own original instrument specifications, the following discussion should be sufficient to make a reasonable start. For others, the Instrument Library File (INSTLIB.I) is provided on the INSMUS-8 Distribution Disk for a variety of "debugged" instruments available for use through the LIB and LIBINS statements.

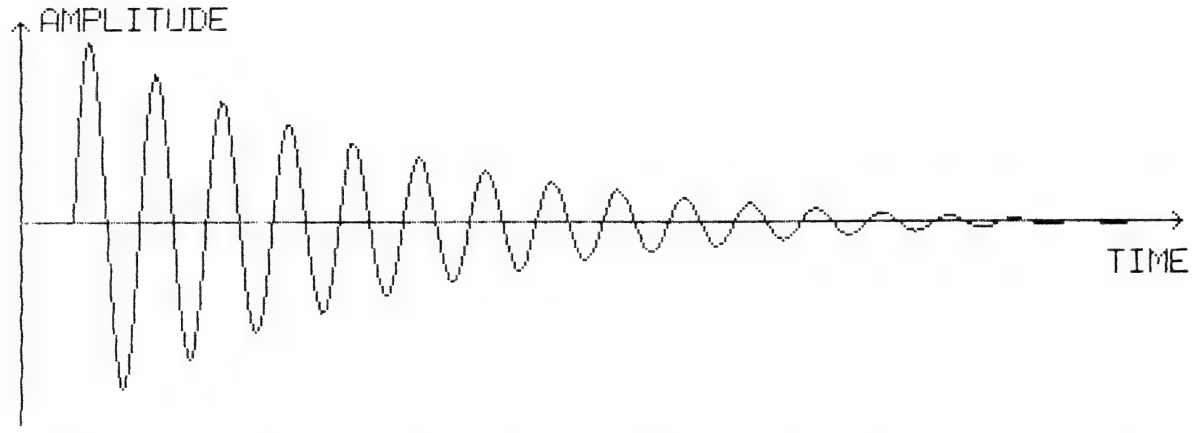
4.1 RELEVANT PHYSICS OF SOUND

Conventional musical instruments create sound either by striking or plucking something, or by exciting a column of air in a closed tube. Electronic instruments utilize oscillators which produce electrical vibrations that eventually drive a loudspeaker. In most cases the sound waveform is approximately periodic with a period inversely proportional to the pitch of the tone. Some instruments, such as a snare drum, however, produce a waveform that is primarily random in nature, while others may have a continuously variable pitch, such as a slide trombone or a violin. While the INSMUS-8 system is very flexible, the latter two varieties of instruments are not specifically provided for. Nevertheless a nearly infinite variety of definite pitch instrumental sounds is possible.

The basic timbre (color) of a tone is determined by its harmonic makeup. Harmonics are component frequencies of the tone that occur at its fundamental frequency (the actual pitch frequency of the musical note being sounded) and integral multiples of that frequency. It is the relative amplitude or loudness among the various harmonics that determines the tone's timbre. A prepoderance of the fundamental and lower harmonics gives a tone "body" or makes it mellow sounding. A preponderance of the higher harmonics tends to make the tone sharp or shrill sounding. Groups of adjacent harmonics that are louder than remaining harmonics can give a variety of throaty or nasal effects. Proper selection of two or three groups of emphasized harmonics can even give the effect of a singing human voice. Details about specific harmonic properties are beyond the scope of this discussion although the Instrument Library can serve as a starting point. Also check the bibliography in Appendix G for a listing of books and publications that would be helpful for further study.

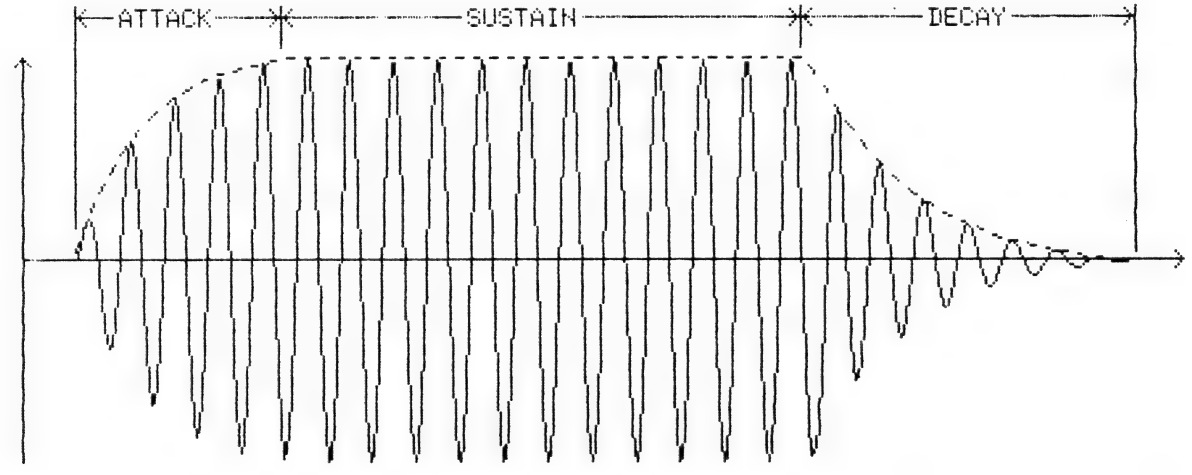
If the tone is absolutely steady, such as holding an organ key down, the waveform will be perfectly periodic, which means that it repeats exactly as long as the key is held down. Such a tone can be synthesized by tabulating just one cycle of its waveform in a waveform table in memory, which is then repeatedly scanned by software. The software then sends the table entries read to a digital-to-analog converter and speaker to be heard. The original music software package provided by MTU in 1977 had the capability of calculating a single waveform table given the desired harmonic composition of the tone and then using that table as the "instrument" with which to play the musical score.

Most of the instruments mentioned above, however, do not produce absolutely steady tones. The loudness (amplitude) of a plucked string instrument, for example, gradually decreases as the string vibrations die away as shown below:



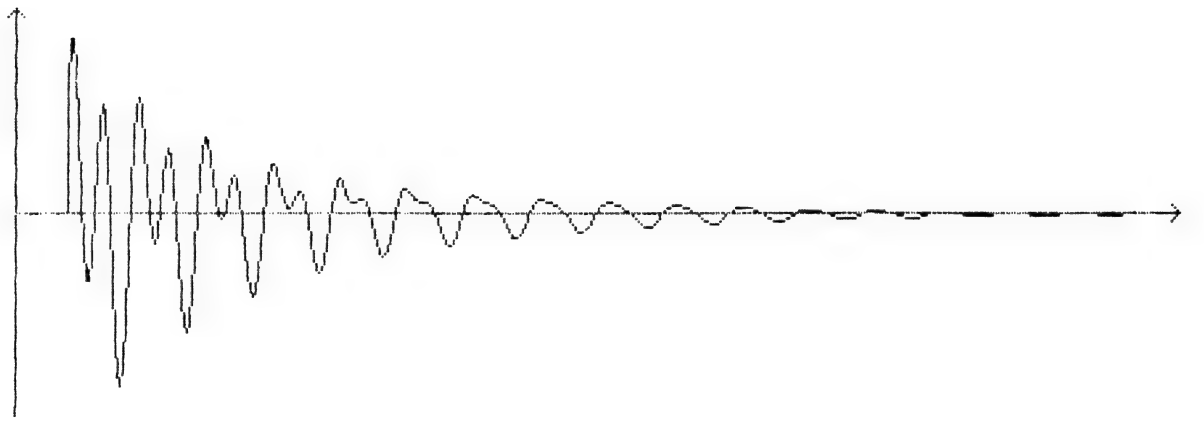
Clearly the scanning of a single waveform table with one cycle of the waveform in it cannot reproduce the decaying characteristic of the plucked waveform.

Blown instruments, such as the trumpet, have a more gradual buildup of loudness and do not start to decay (get softer) until the player's breath pressure is released. The diagram below shows the amplitude contour (known as an envelope) of a typical trumpet note along with names for various parts of the envelope shape:



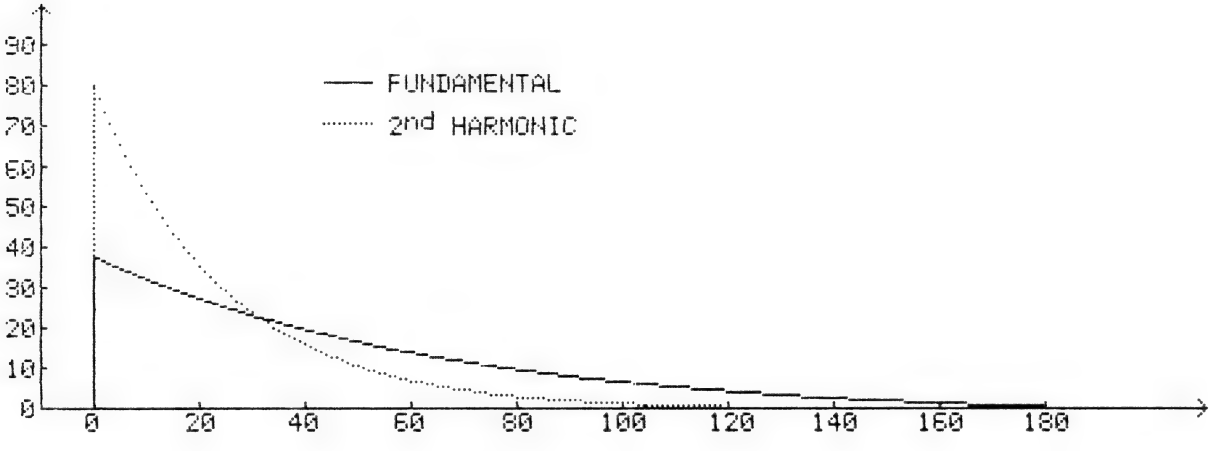
As we shall see later, INSMUS-8 simulates these and other envelope shapes by using a sequence of many waveforms, each calculated with a slightly different amplitude.

For even more realism (or variety) in synthesizing instrument sounds, one has to recognize that even the relative harmonic composition of the tone may vary during a note. Returning to the plucked string example, one would find that a real plucked string would start out with a lot of energy in the higher frequency harmonics. However, since friction with the air is greater for these higher harmonics, they will die out faster than the fundamental and lower harmonics will. The result is a sound that is sharper at the beginning than at the end and an overall waveform that varies in shape as well as amplitude. The drawing below shows a simplified example where the second harmonic starts out stronger than the fundamental but then rapidly dies out leaving the fundamental to complete the tail end of the tone envelope:



This changing of harmonic content during the duration of a note is called "dynamic timbre variation" and is the feature that distinguishes the INSMUS-8 system from nearly all other music synthesizers for microcomputers, both software and hardware. As with amplitude envelopes, INSMUS-8 uses a sequence of slightly different waveform tables to simulate dynamic timbre variation.

A better way to represent tones whose harmonic composition changes over time is to plot the amplitude envelope of <u>each harmonic</u> as a function of time. Thus a drawing of the simplified plucked string would appear as below:



The advantage of this representation is that it tells a lot more about how the tone would actually sound. It is also the form from which instrument specifications are actually prepared.

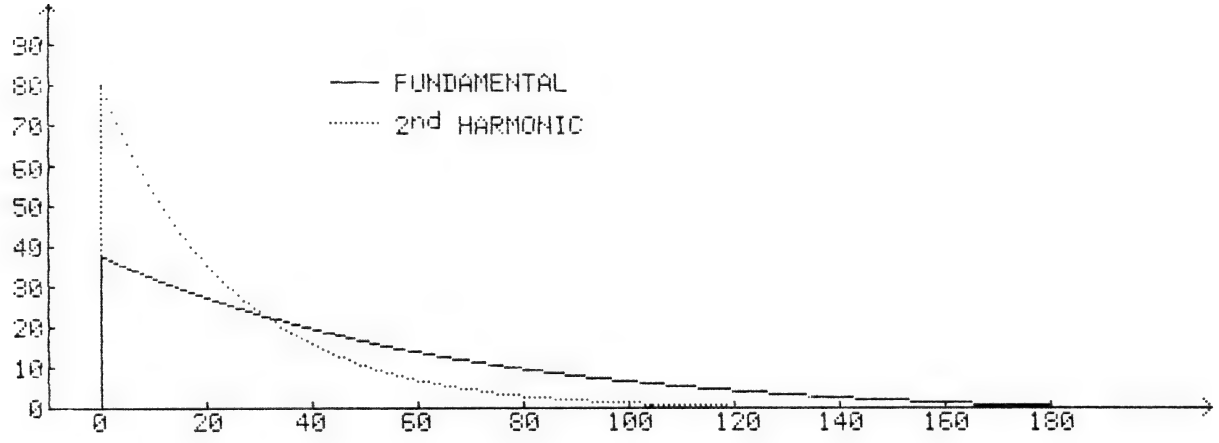
4.2 THE NEWINS STATEMENT

See insert in cover holder

As mentioned above, one way to specify changes in overall amplitude as well as harmonic composition of a tone is to represent the amplitude of each harmonic as a function of time. Data can then be entered into the computer as <u>number pairs</u>, the first number denoting a time position and the second giving the relative amplitude of the harmonic at that time. Imaginary straight lines would connect the specified points to represent the amplitude at intermediate times.

In INSNOTRAN, the NEWINS statement accepts harmonic data in this form and creates all of the internal tables necessary to define the desired sound. Data pairs are interpreted as the endpoints of line segments that would appear on a graph of relative amplitude vs. time. Time values are given in milliseconds, the maximum allowable value being the number of milliseconds equivalent to a whole note (1/1) at the average tempo specified in the NEWINS statement. Relative amplitude is given as a percentage between 0 and 100.

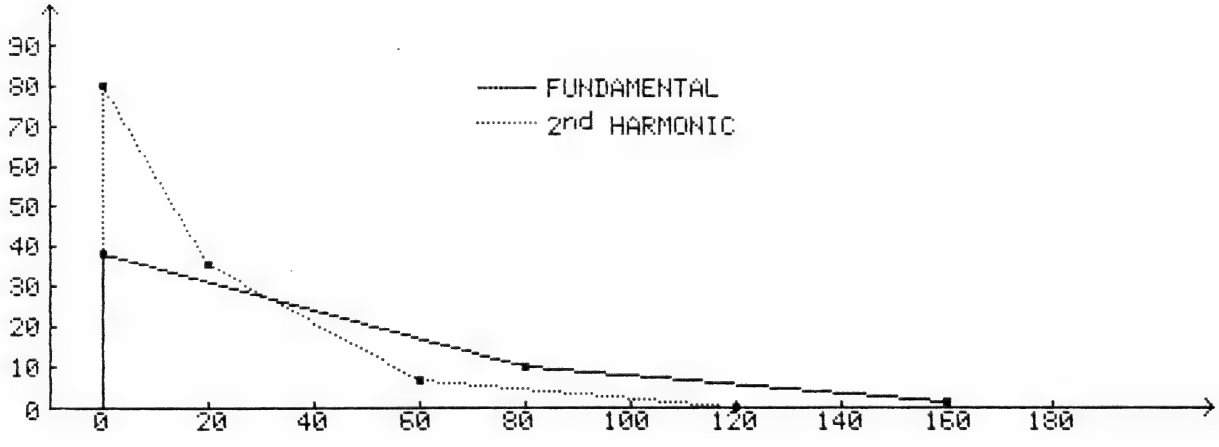
For example, the curves for the two harmonics in the drawing below:



could be represented by the following data in time,amplitude form:

FUNDAMENTAL	2nd HARMONIC
0,38	0,80
80,10	20,36
160,2	60,7
	120,0

This data represents a line segment approximation of the smooth curves shown above. The result of this rather coarse line segment approximation is shown below:



Since only a few endpoints were specified, the approximation is visually fairly coarse but may be entirely satisfactory aurally. Note that the second harmonic curve is specified in more detail than the fundmental (one more point) and at different time values which is allowable since each harmonic is handled separately.

INSMUS-8, being a real-time software synthesizer on a microcomputer, cannot actually reproduce the continuous smooth variations in harmonic content implied by either of the above curves. Instead it plays a sequence of fixed waveforms that differ slightly from each other at a rapid rate. This is exactly how movie film projects continuous motion which it too cannot actually reproduce.

A key issue then is how many different waveforms are used to represent an instrument's sound. A large number will allow a higher "frame rate" and thus produce a better illusion of smooth variation than a small number will. There is a memory limit, of course, so there is as incentive to keep the number of tables used small consistent with acceptable reproduction quality. Using too few tables will make an instrument sound "jerky" just like the low frame rate of old movies. Typically, instrument definitions use 10 to 30 waveforms each. There is room in memory for around 100 waveforms in all, depending on the length of the score.

4.2.2 Example NEWINS Statement

The example instrument illustrated above can be defined by the following NEWINS statement:

```
NEWINS 5 W5 N=18 A=100 1/4=200  H1 0,38;80,10;160,2;  H2 0,80;20,36;60,7;120,0
```

The first number after the keyword NEWINS is the identification number of the instrument which in this case is instrument 5. This is used to refer to this instrument in other statements and must have a value between 1 and 15 inclusive. The W followed by a number is the ID of the waveform set that is used internally in defining the instrument. If you are just using NEWINS statements to define instruments, then it is appropriate to make the waveform set ID the same as the instrument ID which has been done in this example. Later we will see how to create additional, possibly completely different, instrument sounds from the same waveform set thus saving substantial amounts of memory. The N followed by an = and a number is the number of waveform tables to be created in the waveform set and thus used by the instrument. To the person who coded the instrument definition, 18 was enough to give a smooth sound for the relatively simple harmonic envelope curves.

The A followed by an = and a number specifies the overall amplitude (loudness) of the instrument. 100 specifies the maximum without overflow in the digital calculations and is generally preferred to minimize background noise unless it is specifically desired that the particular instrument sound more softly than others used in the score.

The 1/4=200 specifies the typical tempo setting that will be in effect when this instrument plays notes. If the actual tempo setting in effect is indeed equal to that specified, then the millisecond time scale used to specify the harmonic envelopes will be accurate. If the actual tempo is faster or slower, then the envelopes will be compressed or expanded timewise in proportion. Note that this is not a limiting factor; the instrument may be played at any tempo desired. However if the difference becomes too great, the instrument will sound "funny". Typically a + or - variation up to 30% or even more is hardly noticable in the instrument's sound.

The remaining parameters define the line segment endpoints for the harmonic amplitude envelopes. Note that all harmonic numbers not specifically referenced are absent (zero amplitude) in the instrument. Also, time periods not spanned by an amplitude line segment will also have zero amplitude for that harmonic during that time. Note however that the harmonic amplitudes at the latest specified time point will be held from that point on. Thus if the envelope is intended to decay to zero, the last specified time point for each harmonic

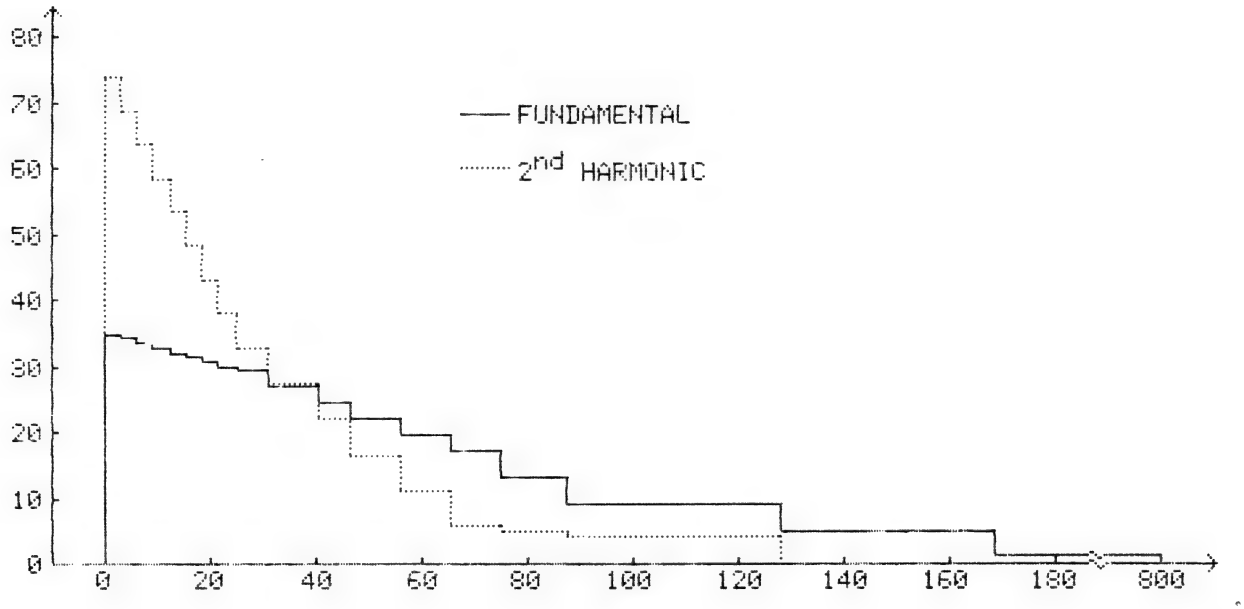
should give a zero amplitude. If there is too much data to fit on one line (which is usually the case), the statement may be split anywhere a semicolon is used and continued on the next line. Note that the continuation must not start in column 1. It is common practice to use blanks liberally to get things to line up on the listing and thus make complex NEWINS statements easier to read.

4.2.3 Waveform Table Usage Optimization

While conventional movie film runs at a constant frame rate regardless of the amount of motion in the scene, the INSMUS-8 system is capable of speeding up the waveform sequence rate when the harmonic envelopes are changing rapidly and slowing it down when they are changing slowly. This allows better use to be made of the typically limited number of waveform tables available. If there is a significant change in amplitude or harmonic content over a given period of time, a relatively large number of waveform tables is needed to provide a smooth transition over that time period. In contrast, a period of time with very little harmonic change will require only a few tables. The NEWINS statement processor in INSNOTRAN optimizes the the use of waveform tables to most smoothly simulate the desired envelopes using the number of tables specified.

The variable "frame rate" described above is actually implemented via another table called a waveform sequence table. This table contains a string of pointers, each of which points to a waveform. The waveform sequence table is scanned at a constant frame rate and contains a large number (256) of entries. During periods of slow change, several successive entries in the sequence table would point to the same waveform while during periods of faster change, fewer would point to a given waveform. As mentioned earlier, NEWINS creates and optimizes this table automatically.

The drawing below shows the actual harmonic envelopes that would be produced by playing a compiled version of the example NEWINS statement above. Note that the width of the steps varies in order to maintain a relatively constant difference in height from step-to-step. INSNOTRAN actually uses a weighted average of step height differences among the active harmonics in its optimization to equalize the "effective auditory" step heights. Also note that since the fundamental did not specifically decay to zero, its last specified amplitude (2 in this case) is extended to the end of the time scale (in this example, a whole note duration at the average tempo which is 800 milliseconds). Finally note that the amplitudes of both harmonics are slightly less than specified. The amplitude optimizer determined that the specified values would overflow (the percentages do add up to more than 100 near the beginning) and so reduced all of the values proportionately just enough to avoid the overflow. If A had been set less than 100%, the amplitudes would have been reduced even more.



4.3 WAVESET AND SEQTAB STATEMENTS

Another way to define an instrument in INSNOTRAN is to use the WAVESET and SEQTAB statements. These statements directly correlate to the old INSMUS Construct Waveform Set and Build Waveform Sequence Table commands. Creating an instrument definition using these statements is a two step process. First, the WAVESET statement is used to create a waveform set. The "time" parameter in the data pairs used by the WAVESET statement is actually a waveform table number; thus the user has more control over the contents of each waveform table. After the waveform set is created, a waveform sequence table containing pointers to waveforms in the set must be created with the SEQTAB statement. The various parameters in the SEQTAB statement allow direct control over the dwell times for each waveform table, alternate sequencing of the waveforms (such as backward!), and even arbitrary selection of individual waveform tables. Additionally, MODIFY and WARBLE statements process waveform sequence tables in specific ways to create common musical effects.

4.3.1 The WAVESET Statement

The WAVESET statement is very similar to the NEWINS statement described in section 4.2. The main difference is that the "time" axis is no longer calibrated in *milliseconds*, instead it is calibrated in "waveform tables". Allowable values are from 0 to N-1 inclusive where N is the total number of waveform tables specified for the statement. Amplitude values are specified in the same way as NEWINS: as relative values between 0 and 100. Below is an example WAVESET statement:

```
WAVESET W3 N=20 A=75  H1 0,10; 5,30; 9,65; 14,50; 19,0;
                      H2 0,30; 3,60; 5,87; 14,78; 19,0;
                      H3 0,50; 2,75; 3,85;  5,95; 14,88; 19,0;
                      H4 0,35; 2,50; 4,65; 10,40; 14,30; 17,0;
```

The "W3" argument identifies the waveform set created by the statement as waveform set #3. This number is used to refer to the set in succeeding SEQTAB statements. Up to 15 different waveform sets may be in memory at once so the W argument is limited to values between 1 and 15. It is allowable for a WAVESET statement to use the same ID as an earlier WAVESET statement in order to redefine instruments in mid-song. The only restriction is that the new set may

not contain any more waveforms than the previous set. The "N=20" argument specifies the number of waveform tables to be built. The same considerations apply in selecting it as in the NEWINS statement. Note however that you cannot just arbitrarily change the N value in an existing statement without changing anything else as in NEWINS because the harmonic specifications are in terms of waveform table number which must be between 0 and N-1. The "A=75" argument specifies that all of the amplitudes are to be scaled to 75% of their maximum non-overflow value.

The line segment specifications are formatted just like in the NEWINS statement. The time values however are waveform numbers between 0 and N-1 inclusive. Waveform tables not spanned by a harmonic's line segment will contain zero amplitude for that harmonic except at the end where the last specified amplitude will be carried throughout the remaining tables. Note that if you wish to explicitly specify the exact harmonic makeup of each table (i.e., pointwise, perhaps directly from a sound analysis readout), simply specify 0, 1, 2, 3, ... N-3, N-2, N-1 for the waveform numbers. Be sure to explicitly specify zero amplitude when needed for any harmonic that had an earlier non-zero amplitude.

4.3.2 The SEQTAB Statement

After a waveform set has been created, a waveform sequence table must be created before the waveforms can be "played". A waveform sequence table always contains 256 "pointers" to waveforms in waveform sets. When a note is played, INSPLAY scans through the waveform sequence table playing the waveforms the pointers point to. This scan is at such a rate that the whole 256 entry table is scanned in one whole note's duration. The amount of time spent at one position of the sequence table is therefore a 1/256 note's duration which is defined as one tempo period. The sequence table scan always begins at the beginning and stops when the note ends. Thus a quarter note will only scan through the first 64 pointers before the note ends. Since it would be inconvenient to specify each of the 256 sequence table entries, the SEQTAB statement is provided to automatically fill the table according to user specifications.

4.3.2.1 Simple Stretch

Several "models" for an instrument's envelope are available in the SEQTAB statement. The simplest is the simple stretch. The simple stretch option, which is specified by X=0, is most useful for struck and plucked string instruments where the envelope has an essentially instantaneous attack, rapid initial decay, and slower final decay. Its purpose is to simply warp the time scale of the waveform set such that waveforms are sequenced rapidly at the beginning and more slowly at the end. An example SEQTAB statement using simple stretch is shown below:

```
SEQTAB  3  X=0  BR=01  W2:0,15
```

The "3" is the ID for the sequence table to be created thus this table will be referred to as instrument #3 in subsequent ASSIGN statements. The "X=0" selects the simple stretch algorithm. The value assigned to BR is a pair of hexadecimal digits. The first digit (0 in this case) is the B (block size) parameter and the second digit (1 in this case) is the R (repeat) parameter. Together they specify the degree of stretch to be used. The "W2:0,15" specifies that waveforms 0 through 15 from waveform set #2 are to be used in creating the sequence table.

As mentioned above, B and R specify the degree of stretch. Basically, B+1 specifies the initial "block" length and R+1 specifies the number of times a block length is to be repeated before being incremented by 1. In different language, the block length is the number of times a given waveform reference is to be repeated in the sequence table prior to switching over to the next waveform. Its initial value is B+1 so it can range from 1 to 16. The block repeat value determines how many times a particular block length is to be used on successive waveform references before the block length is incremented by 1. The actual number of repeats is R+1 which also has a range from 1 to 16. A little bit of mathematical analysis will reveal that this algorithm implements quadratic stretching. If the waveform set is exhausted before the waveform sequence table is filled, the remaining entries are filled with repetitions of the last waveform (#15 in this case). If the table is filled before all of the waveforms are used, the process simply terminates without an error indication.

Perhaps the best way to understand the stretching operation is to look at what the example statement given earlier actually does. The resulting sequence table would look like:

```
  0   00 01 02 02 03 03 04 04 04 05 05 05 06 06 06 06 07 07 07 07 08 08 08 08 08
 25   09 09 09 09 09 10 10 10 10 10 10 11 11 11 11 11 11 12 12 12 12 12 12 12 13
 50   13 13 13 13 13 13 14 14 14 14 14 14 14 14 15 15 15 15 15 15 15 15 15 15 15
 75   15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
100   15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
125   15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
150   15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
175   15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
200   15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
225   15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
250   15 15 15 15 15 15
```

By looking at the resulting table, we can see that since the initial block length (B+1) is 1 that the first waveform is used just once. Since the repeat value (R+1) is two, the next waveform is also used once. Then the block length is incremented so that waveform 2 is used twice and waveform 3 is used twice. Then another increment is executed and the next two waves are used three times each and so on. Since the waveform set was exhausted before the table was complete, it was filled out with references to the last waveform.

It is entirely legitimate to create more than one waveform sequence table from the same waveform set. For example, the overall shape of the harmonic envelopes for members of the violin family played pizzicato (plucked) are all similar but the envelopes are longer for the larger members of the family. You could generate just one waveform set to cover all of them and then produce several sequence tables referring to it, each with different values of B and R. You could even specify "W2:15,0" in which case waveforms would be used from the set in reverse order giving a strange effect indeed.

4.3.2.2 Multi-Part Sequence Table Construction

While the simple stretch just described is ideal for instrument sounds with rapid or instantaneous attacks immediately followed by an exponential decay, it is not suitable for sustained wind instrument sounds such as a trumpet, clarinet, etc. By setting the X parameter in the statement non-zero, other sequence table construction options are enabled.

The simpler case is when X is an odd number (1, 3, 5, . . . , 15) which specifies that distinct attack, sustain, and release phases be recognized when the sequence table in constructed. The attack phase is built starting at the beginning of the sequence table using the first group of B, R, and W parameters in exactly the same manner as the simple stretch just discussed. However when the waveform set is exhausted, the attack phase is declared to be complete and sequencing for the sustain phase is started. The length of the sustain phase is proportional to the value of X according to the expression: 32*INT(X/2). Thus if X is equal to 1 there is no sustain phase and processing immediately proceeds to the release phase. If there is a finite length sustain phase, the last waveform used during the attack phase is simply entered into the table 32*INT(X/2) times. After the sustain phase, the release phase is constructed using the second group of B, R, and W parameters in a manner similar to the attack phase. For many wind instruments it is acceptable to simply specify the reverse of the same waveform set that was used for the attack. If the release waveform set is exhausted, the remainder of the table is filled with the last waveform as before. If the table becomes prematurely filled at any point, the command is declared complete without any error indication.

As before, an example should serve to illustrate the concepts better than words:

```
SEQTAB  4  X=9   BR=10 W4:1,9   BR=21 W5:0,7
```

Here, the sequence table for instrument #4 is being constructed. X=9 which gives a three part envelope with level sustain. The attack will be constructed from waveforms in set #4 starting with the second waveform (#1) and going through the tenth (#9). B=1 and R=0 (effective values 1 and 2) for the attack. The sustain will be waveform #9 from this set and will be 128 entries long or equivalent to a half-note duration. The decay will be built from the first 8 waveforms in set #5 with B=2 and R=1. The actual table this statement generates is:

```
  0  01 01 02 02 02 03 03 03 03 04 04 04 04 04 05 05 05 05 05 05 06 06 06 06 06
 25  06 06 07 07 07 07 07 07 07 07 08 08 08 08 08 08 08 08 08 08 09 09 09 09 09 09
 50  09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09
 75  09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09
100  09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09
125  09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09
150  09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09 09
175  09 09 09 09 09 09 09 00 00 00 01 01 01 02 02 02 02 03 03 03 03 04 04 04 04
200  04 05 05 05 05 05 06 06 06 06 06 06 07 07 07 07 07 07 07 07 07 07 07 07 07 07
225  07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07
250  07 07 07 07 07 07
```

Note that starting at entry 182, the table entries point to waveforms in set #5 instead of set #4. This is indicated by italics.

When X is an even number (except zero), the attack and release phases are constructed in exactly the same way as when X is odd. However rather than having a perfectly flat sustain phase, a tremolo effect (a better name might be "warble") is added to the sustain phase. This is accomplished by cycling through the last three waveforms in the attack set during the sustain phase which is 16X table entries long. A complete warble cycle is 8 table entries long and for a typical Tempo setting of 1/4=750, the warble rate is about 10.6 hertz. When constructing the waveform set whose last three members will be used for the warble, it is important that there be enough contrast among them to be heard but not so much that the effect becomes exaggerated or table switching noise becomes excessive. A good rule of thumb is to use an amplitude contrast of about 10% coupled with a slight timbre contrast.

The above example statement modified for a warbled sustain would be:

```
SEQTAB  4  X=8   BR=10 W4:1,9   BR=21 W5:0,7
```

and the resulting sequence table would be:

```
  0  01 01 02 02 02 03 03 03 03 04 04 04 04 04 05 05 05 05 05 05 06 06 06 06 06
 25  06 06 07 07 07 07 07 07 07 07 08 08 08 08 08 08 08 08 08 09 09 09 09 09 09
 50  09 09 09 09 08 08 07 07 08 08 09 09 08 08 07 07 08 08 09 09 08 08 07 07 08
 75  08 09 09 08 08 07 07 08 08 09 09 08 08 07 07 08 08 09 09 08 08 07 07 08 08
100  09 09 08 08 07 07 08 08 09 09 08 08 07 07 08 08 09 09 08 08 07 07 08 08 09
125  09 08 08 07 07 08 08 09 09 08 08 07 07 08 08 09 09 08 08 07 07 08 08 09 09
150  08 08 07 07 08 08 09 09 08 08 07 07 08 08 09 09 08 08 07 07 08 08 09 09 08
175  08 07 07 08 08 09 09 00 00 00 01 01 01 02 02 02 02 03 03 03 03 04 04 04 04
200  04 05 05 05 05 05 06 06 06 06 06 06 07 07 07 07 07 07 07 07 07 07 07 07 07
225  07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07
250  07 07 07 07 07 07
```

4.3.3 The MODIFY Statement

This statement is intended to be used with plucked or struck type instruments to create a strumming sound. Rather than creating a waveform sequence table from scratch, it modifies a "source" table and generates a "destination" table that in fact might be the same table as the source. Fundamental operation is very simple: in effect the first N entries in the source table are repeatedly copied into the destination table until the destination table is filled.

A typical example statement might be:

```
MODIFY 4=3 1/32
```

Here, sequence table #4 will be created from sequence table #3. The number of entries in table #3 that will be duplicated in table #4 for each duplication cycle is that number that corresponds to a 1/32nd note duration which is 8. If the tempo is 1/4=750, then the repeat period will be 93.75 milliseconds or 10.6 repeats per second. Although here the original sequence table is preserved, it is permissable to make the destination table equal the source table and thus replace it. The resulting table would look like this (see section 4.3.2.1 for a listing of the source table):

```
  0  00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00
 25  01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01
 50  02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01 02
 75  02 03 03 04 04 00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01 02 02
100  03 03 04 04 00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01 02 02 03
125  03 04 04 00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01 02 02 03 03
150  04 04 00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01 02 02 03 03 04
175  04 00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04
200  00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00
225  01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01 02 02 03 03 04 04 00 01
250  02 02 03 03 04 04
```

Like MODIFY, the WARBLE statement is used to construct a new waveform sequence table by processing the contents of an existing table. The audible effect of this processing is a "warbling" of the sound represented by the source sequence table and its associated waveforms. The modification works by first copying M entries from the source table into the destination table and then copying N entries backward from the current position in the source into the destination where M and N are specified in the statement. This completes one "warble cycle" in the destination table. The process is then repeated for additional cycles starting at the current position in each table until the destination table is filled. Typically M and N will be the same which means that the first part of the source instrument's envelope is repeated forward and backward in the destination instrument's definition. If M is greater than N, the warbled envelope "crawls" along the envelope of the source. N being greater than M is not allowed. M and N are given in actual waveform table counts. If you wish for the warble repetition period to have a specific musical duration (note fraction), simply express that desired duration in 1/256th notes and make M+N equal to that number.

Below is an example of a WARBLE statement:

```
WARBLE 5=3 9 7
```

Here the source sequence table is number 3 and the table being created is number 5 thus preserving the source. M=9 and N=7 so the destination will be created by copying 9 entries from the source forward and then copying 7 entries from the source backward and repeating thus advancing a net of 2 entries per warble cycle. The total warble cycle will be 16 entries long which is a 1/16 note duration independent of the tempo setting (i.e., the warble rate follows the current tempo). The table resulting from this statement would look like this (see section 4.3.2.1 for a listing of the source table):

```
  0   00 01 02 02 03 03 04 04 04 04 04 04 03 03 02 02 02 02 03 03 04 04 04 05 05
 25   05 05 04 04 04 03 03 03 03 04 04 04 05 05 05 06 06 05 05 05 04 04 04 04 04
 50   04 05 05 05 06 06 06 06 06 06 05 05 05 04 04 05 05 05 06 06 06 06 07 07 06
 75   06 06 06 05 05 05 05 06 06 06 06 07 07 07 07 07 07 06 06 06 06 06 06 06 06
100   07 07 07 07 08 08 07 07 07 07 06 06 06 06 07 07 07 07 08 08 08 08 08 08 07
125   07 07 07 07 07 07 07 08 08 08 08 08 08 08 08 08 08 08 07 07 07 07 08 08 08 08
150   08 09 09 09 09 08 08 08 08 08 08 08 08 08 08 08 09 09 09 09 09 09 09 09 08 08
175   08 08 08 08 09 09 09 09 09 10 10 09 09 09 09 09 08 08 09 09 09 09 09 10 10
200   10 10 10 10 09 09 09 09 09 09 09 09 10 10 10 10 10 10 10 10 10 10 09 09 09
225   09 10 10 10 10 10 10 11 11 10 10 10 10 10 10 10 10 10 10 10 10 10 11 11 11 11
250   11 11 10 10 10 10
```

4.4 USING AN INSTRUMENT LIBRARY

Instrument Library Files contain copies of the INSNOTRAN statements used to define each instrument in the library. These statements are placed into the source code stream by a LIBINS statement. Specific parameters, such as instrument ID numbers and waveform table numbers, are inserted by the INSNOTRAN compiler according to the LIBINS statement. Instrument definitions are stored in a library file by using the MTU text editor, or some other means of entering ASCII files, following the format detailed below.

An Instrument Library File is stored simply as a string of ASCII characters; thus, it can be printed, displayed, or edited just like any other text file. Each instrument definition begins with an ampersand (&), followed immediately by the name of the instrument being defined. The instrument name may contain up to 8 alphanumeric characters. The name is terminated by a blank or a carriage return and is followed by the definition statements.

The definition statements are in standard INSNOTRAN format. Only statements used to define a new instrument (NEWINS, WAVESET, QWAVESET, SEQTAB, QSEQTAB, MODIFY, WARBLE) or comment statements are allowed; other commands will generate an error and will be ignored. All Instrument and Waveform Set ID numbers found in these statements will be supplied by the LIBINS statement making the definition. There should be two characters in the place of Instrument or Waveform Set ID's in the Library File statements. These characters are merely place-holders; they can be anything except blanks (underline, _, is recommended).

When a library definition is called by a LIBINS statement, all of the statements, except the first one with the name, will be printed in the listing along with the object code they generate, just as if they were a part of the regular source code file. They are not assigned statement numbers, however, so you can distinguish statements generated by a library call from those present in the original source code. The first statement, containing the name and optional comments, is not printed because it has no meaning as an INSNOTRAN statement.

The following are examples of valid entries in an INSNOTRAN Instrument Library File:

```
&REED1  This line will not be printed on the listing.
*  Reed instrument used in "The Dance of the Reed Flutes"
*  32 Waveform Tables used.
*  Optimum Tempo:  1/4 = 300
*  Pitch Range: No lower limit, high limit is C#5 because of aliasing.
WAVESET W__ N=32 H1 0,0; 4,50; 16,100; 32,50;
                 H2 0,15; 16,0; 24,65; 32,25;
                 H3 0,50; 8,65; 16,25; 32,0;
                 H4 0,50; 3,0; 8,0; 16,50; 32,0;
                 H5 0,50; 6,25; 16,50; 32,12;
                 H6 0,50; 16,25; 25,65; 32,12;
                 H7 0,0; 8,50; 32,0
SEQTAB __ X=0 BR=04 W__:0,32
&STRBANJO
*  Banjo-like instrument with "strumming" effect.
*  32 Waveforms used.
*  Pitch Range:  Useful from G3 to G5.
NEWINS __ W__ N=32 1/4=500 H1 0,40; 250,30; 750,20; 1500,10; 1750,0;
                   H2 0,60; 250,45; 750,20; 1500,0;
                   H3 0,80; 250,50; 750,20; 1000,0;
                   H4 0,90; 250,50; 750,0;
                   H5 0,100; 500,0
MODIFY __=__ 1/16
```

A LIBINS statement using the first library entry might look like this:

```
LIBINS  5  W5  REED1
```

INSNOTRAN would then search the disk file specified by a previous LIB statement looking for the &REED1 name line. Once it is found, succeeding lines are copied from the file into the score stream until the name line for the next instrument or end of file is reached. Whenever a W parameter (waveform set ID number) is copied, the W parameter in the LIBINS statement is substituted instead. Likewise, whenever an instrument ID is copied, the the ID specified in the LIBINS statement is substituted. Thus in this example, the following statements would be inserted into the source stream as a result of the above LIBINS statement:

```
*  Reed instrument used in "The Dance of the Reed Flutes"
*  32 Waveform Tables used.
*  Optimum Tempo:  1/4 = 300
*  Pitch Range: No lower limit, high limit is C#5 because of aliasing.
WAVESET W 5 N=32 H1 0,0; 4,50; 16,100; 32,50;
                 H2 0,15; 16,0; 24,65; 32,25;
                 H3 0,50; 8,65; 16,25; 32,0;
                 H4 0,50; 3,0; 8,0; 16,50; 32,0;
                 H5 0,50; 6,25; 16,50; 32,12;
                 H6 0,50; 16,25; 25,65; 32,12;
                 H7 0,0; 8,50; 32,0
SEQTAB  5 X=0 BR=04 W 5:0,32
```

Operation is similar if the second instrument had been called instead. Note however that the MODIFY statement in the definition will always have the destination waveform sequence table equal to the source table thus overwriting it. One limitation of the library facility is that there is no convenient way to create multiple waveform sequence tables (instrument) from a single waveform set. In such a situation, it will be necessary to store just the WAVESET (or QWAVESET) statement in the library and then specifically insert the needed SEQTAB, MODIFY, etc. statements directly in the score.

4.4.2 Using and Creating the QWAVESET Statement

Presumably the instrument definitions in a library would be thoroughly "debugged", i.e., they would not change from compilation to compilation. The standard WAVESET and NEWINS statements may require compilation times of several seconds to a minute or more for amplitude and table use optimization. QWAVESET provides a way to insert previously compiled object code from a WAVESET statement into the source stream while still allowing the parameter substitution afforded by the LIBINS statement. Note that QWAVESET only corresponds to the WAVESET statement; a NEWINS statement would have to be recoded into a QWAVESET and a QSEQTAB statement as described later.

In order to create a QWAVESET statement from a WAVESET statement, the first step is to compile the WAVESET statement so that the generated object code can be examined. This may have already been done in a previous compilation or a special INSNOTRAN score may be created just to compile the WAVESET statement. Below is an example of such a score for compiling a single WAVESET statement:

```
* TEST CASE FOR WAVESET COMMAND
*
WAVESET W12 N=32 A=75  H1 0,100; 31,0;  H2 0,80; 29,0;
                       H3 0,64; 25,0;   H4 0,48; 21,0;
                       H5 0,32; 16,0;   H6 0,16; 8,0;
                       H7 0,0; 8,25; 31,0;
                       H8 3,0; 10,20; 31,10
ENDCMD
END
```

The listing produced by INSNOTRAN when the above score is compiled looks like this:

```
INSTRUMENT SYNTHESIS NOTRAN MUSIC COMPILER

--- COMMANDS SECTION ---

STMT LOCN OBJECT - BYTES  SOURCE STATEMENT
1                         * TEST CASE FOR WAVESET COMMAND
2                         *
3    2000 EE 00 F5 02 21  WAVESET W12 N=32 A=75  H1 0,100; 31,0;  H2 0,80; 29,0;
4    2005                                        H3 0,64; 25,0;   H4 0,48; 21,0;
5    2005                                        H5 0,32; 16,0;   H6 0,16; 8,0;
6    2005                                        H7 0,0; 8,25; 31,0;
7    2005 01 00 4D 1F 00                         H8 3,0; 10,20; 31,10
     200A FF 02 00 3D 1D
     200F 00 FF 03 00 31
     2014 19 00 FF 04 00
     2019 24 15 00 FF 05
     201E 00 18 10 00 FF
     2023 06 00 0C 08 00
     2028 FF 07 00 00 08
     202D 13 1F 00 FF 08
     2032 03 00 0A 0F 1F
     2037 07 FF 00
8    203A FF              ENDCMD

--- NOTES SECTION ---

STMT LOCN OBJECT - BYTES  SOURCE STATEMENT
9    203B 00              END
```

It is the hexadecimal object code to the left of and below the WAVESET source statement in the listing that is of interest in creating a QWAVESET that corresponds to the given WAVESET statement. Each pair of hex digits represents one byte of object code.

To construct the QWAVESET statement, first enter the QWAVESET keyword, the waveset number (W parameter), and the number of waveform tables (N parameter) which must be equal to the N parameter in the original WAVESET statement. Note that there is no A parameter (it will be the same as the original WAVESET) since the amplitudes are part of the object code. Following this you simply copy object code bytes from the listing. Normally The first 3 bytes of of the object code generated by the WAVESET statement are not significant for constructing the corresponding QWAVESET statement (the represent the opcode, W, and N parameters). However if the statement is the very first in the score (as it is here), two additional bytes are created which should also be ignored. The key

is to find the opcode byte (the F5) and skip it and the next two bytes before starting. Bytes are copied from the listing until the end of the code generated for the WAVESET statement. Usually there are more than will fit on a line in which case simply follow the last byte with a semicolon and continue on the next line in column 2 or greater. Thus the QWAVESET statement corresponding to the above WAVESET statement would be:

```
QWAVESET W12 N=32  01 00 4D 1F 00 FF 02 00 3D 1D 00 FF 03 00 31 19 00 FF 04 00;
                   24 15 00 FF 05 00 18 10 00 FF 06 00 0C 08 00 FF 07 00 00 08;
                   13 1F 00 FF 08 03 00 0A 0F 1F 07 FF 00
```

For complex WAVESET statements, it may be easier (and more accurate) to use the MTU editor to simply delete all of the unnecessary text in the listing of the WAVESET statement and add the parameters needed for the QWAVESET statement.

4.4.3 Converting NEWINS into QWAVESET and QSEQTAB

The NEWINS statement actually compiles into object code representing a WAVESET and a special kind of sequence table. Thus the quick version of a NEWINS statement will consist of a QWAVESET followed by a QSEQTAB statement. Below is the INSNOTRAN listing of a NEWINS statement:

```
STMT LOCN OBJECT - BYTES  SOURCE STATEMENT
1                         * EXAMPLE NEWINS STATEMENT
2    2000 EE 00 F5 02 13  NEWINS 5 W5 N=18 A=100 1/4=200  H1 0,38;80,10;160,2;
3    2005 01 00 59 08 4B                                  H2 0,80;20,36;60,7;120,0
     200A 0D 2C 0F 17 10
     200F 0D 11 04 FF 02
     2014 00 BD 08 54 0D
     2019 0F 0F 0B 10 00
     201E FF 00
     2020 F8 05 02 01 03
     2025 01 04 01 05 01
     202A 06 01 07 01 08
     202F 01 09 01 0A 02
     2034 0B 03 0C 02 0D
     2039 03 0E 03 0F 03
     203E 10 04 11 0D 12
     2043 0D 13 01 FF
4    2047
5    2047 E3 05           DRAWINS 5
6    2049 E2 05           DRAWENV 5
```

The first part of the object code is converted into a QWAVESET statement much like before. First find the F5 op code byte and skip it and the next two bytes. Next, copy the pairs of hexadecimal digits into the QWAVESET statement as before but stop when you have copied the sequence *FF 00*.

The remaining bytes will be used in a QSEQTAB statement. Create the QSEQTAB statement by entering the keyword QSEQTAB, the instrument number, and the waveset number (W parameter) which must be equal to the waveset number of the preceeding QWAVESET. Following this you simply copy the remaining object code bytes from the listing starting 2 bytes beyond the *FF 00* encountered earlier (i.e., skip the F8 op code and the next byte). Continue copying until the end of the NEWINS statement object code (the last byte copied will be FF). The QWAVESET and QSEQTAB equivalent to the preceeding NEWINS statement is shown below:

```
QWAVESET W5 N=18   01 00 59 08 4B;
                   0D 2C 0F 17 10;
                   0D 11 04 FF 02;
                   00 BD 08 54 0D;
                   0F 0F 0B 10 00;
                   FF 00
QSEQTAB 5 W5       02 01 03;
                   01 04 01 05 01;
                   06 01 07 01 08;
                   01 09 01 0A 02;
                   0B 03 0C 02 0D;
                   03 0E 03 0F 03;
                   10 04 11 0D 12;
                   0D 13 01 FF
```

4.5 OPTIMIZING TONE QUALITY

Like everything else in this world, the digital synthesis technique used by the MTU-130/140 and INSMUS-8 is not perfect. For example, only 8772 sound wave points are computed every second and sent to the D-to-A converter which means that frequencies only up to about 3.5KHz may be synthesized. Also, round-off error is introduced into the waveform points due to shortcuts in the computation (needed for speed) and the limited resolution of 8 bit D-to-A conversion. This error gives rise to some residual background noise and distortion in the reproduced sound. For best results, it is necessary to be cognizant of these limitations and utilize techniques to circumvent them or minimize their effect.

4.5.1 Avoiding Alias Distortion

The easiest trap to fall into when using digital sound synthesis is trying to generate frequencies that are too high. There is no lower limit so you can shake the room with bass chords but if one of the harmonic components of a tone is too high in frequency, it is "transformed" into an incorrect frequency rather than simply being lost. At worst the result is severe distortion while at best, strange sounds are produced.

Whenever a simple sine wave tone is digitally synthesized, there are actually two tones produced. One of these is at the correct frequency of F Hz while the other, which is called an alias, is 8772-F Hz. Remember that this action applies to each harmonic of each tone individually. When F is fairly small, such as 1000, the alias at 7772 is widely separated from it and the low-pass filter in the computer's D-to-A converter can completely separate them and reject the alias tone. Tones as high as 3500Hz can be separated adequately from their aliases (5272Hz) by this filter. If you try to synthesize a very high frequency, such as 7000Hz, then the correct frequency will actually be blocked by the filter but its alias at 8772-7000=1772Hz will pass through and be heard as the wrong frequency. The only way to avoid the resulting alias distortion is to avoid generating frequencies higher than 3500Hz.

As an example of how it is very easy to get into trouble with alias distortion, consider the case of playing C5, which is about 522Hz, with a voice that has been assigned to Instrument 1 as defined below:

```
NEWINS 1 W1 N=20 A=100 1/4=750  H1 0,100; 500,25; 1000,0;
                                H2 0,75;  500,20; 1000,0;
                                H3 0,30;  500,10; 1000,0;
                                H4 0,20;  500,7;  1000,0;
                                H5 0,35;  500,10; 1000,0;
                                H6 0,70;  500,20; 1000,0;
                                H7 0,50;  500,17; 1000,0;
                                H8 0,30;  500,12; 1000,0
```

The tone will consist of the fundamental frequency, 522Hz, the second harmonic at 1044Hz, the third harmonic at 13132z, the fourth at 2088Hz, fifth at 2610Hz, 6th at 3132Hz, 7th at 3654, and eighth harmonic at 4176Hz. The sixth is OK. The seventh harmonic is close to the recommended 3500Hz limit but will probably be OK. The alias of the eighth harmonic however will be at 4596Hz which too close to its 4176Hz to be separated. The highest note that you should attempt to play with this instrument is approximately A4 (440Hz).

While the preceeding example was a boarderline case that may sometimes be acceptable, a more flagarent violation would be playing C6 (1044Hz) with this instrument. Here not just one but four of its harmonics exceed the 3500Hz limit, the highest by so much that its alias is at 420Hz. This tone can be expected to be severely distorted.

On the other hand, playing very low notes with waveforms designed for high notes (and consequently having only a few harmonics) will give an excessively muffled sound. It is desirable in such cases to define new instruments having more harmonics for use on low notes. Likewise, when very high notes are needed such as C5 and beyond, you need to define new instruments having fewer harmonics to avoid alias distortion. In the instrument library, comments give the recommended pitch range in each instrument definition. Appendix I has a table of note frequencies and recommended highest harmonic to aid in avoiding this problem.

The rule then is that the highest harmonic frequency of the highest note in the score that plays it should not exceed 3500Hz if alias distortion is to be avoided. This is not a hard and fast limit so your ears should be the judge. If high harmonics above the limit have low amplitude, as is often the case, then violating the limit by as much as 1000Hz may not be noticable, particularly if several voices are playing.

4.5.2 Minimizing Background Noise

Unlike alias distortion, background noise is an inescapable result of round-off error in digital synthesis and can only be minimized. In advanced (and expensive) digital synthesis systems, DACs with greater than 8 bits are used and "delayed playback", as opposed to "real time" software can be used to reduce these errors well below the threshold of audibility. In the INSMUS-8 system, the most effective way to minimize background noise is to make sure that the maximum allowable amplitude is used in all instrument definitions. This is accomplished by using A=100 in NEWINS and WAVESET statements or omitting the A parameter altogether.

Randomizing the phases of the harmonics is not possible in the INSMUS-8 system as it is in SNOTRAN because the phases are all fixed at +90 degrees. This was necessary to insure that discontinuities do not exist when switching from one waveform table to another while the instrument is playing.

It is also helpful if more than one instrument is playing at once and the instruments themselves have a bright timbre (i.e., a lot of high harmonics consistent with alias limitations). Also, constantly varying harmonic envelopes as opposed to long, constant sustains makes it less likely that the ear can "focus" on a tone and perceive the round-off defects.

5. USING THE TEMPERAMENT STATEMENT

The INSMUS-8 SYSTEM provides a default table of 98 notes which extend from a very low B (one half step below C0) having a frequency of 15.43 hz., up to C8 at 4186.01 hz. The basis for this note table is the 12-tone scale of equal temperament. This scale is the basis for the tuning of most modern musical instruments, and can be used for nearly all music in the European/American tradition. The frequency of each note differs from that of its neighbors by a constant multiplicative factor (approximately 1.05945) which is the twelfth root of two. The chief advantage of the equal tempered scale is that all twelve possible 7-tone diatonic scales (do, re, mi, etc.) based upon the twelve different tones sound equally good (or equally bad, if you're a harmonic purist), and transposition between keys is easy. This results in a chromatic flexibility which is indispensable to modern music.

A drawback to equal temperament not generally appreciated, however, is that the intervals of the equal tempered scale are not identical with the truly harmonic intervals based on the ratios of small whole numbers. The listener with a trained ear can easily perceive the beats resulting from this approximation when two or more notes are played together in harmonic passages. These beats are used to advantage by the skilled piano tuner who makes use of them to tune an instrument to equal temperament by ear alone. Most microcomputer music systems lack sufficient precision in specifying frequency to be really concerned about about the relatively small errors involved (i.e., the system tuning error may exceed the error associated with equal temperament). The INSMUS-8 system however has an inherent error of only $\pm$.06Hz which is much smaller than the temperament errors being considered here.

Other scales could be used as the base for the note table if the user were interested in harmonic perfection, historical or cultural accuracy, or avant-garde microtonal compositions. Two different methods have been provided in the INSMUS-8 system which allow the user to do just this. One method loads note tables based upon data supplied with the software package and the other allows complete user specification of both the number and size of all the intervals in the octave. Both methods use the TEMPERMNT statement in INSNOTRAN or the $FA opcode.

5.1 USING THE PREDEFINED TEMPERAMENTS

Five different scales are supplied with INSMUS-8. The first has been discussed above and is normally used as the default basis for the note table on startup. Four additional tunings are supplied, all generate note tables based upon 12-tone scales, and the standard method of denoting pitches in the Notes Section of the INSNOTRAN score applies. The following are brief descriptions of the predefined temperaments supplied with INSMUS-8.

5.1.1 TEMPRMNT 1 (Pythagorean)

This tuning, attributed to the Greek mathematician and philosopher Pythagoras, is based upon the "circle of fifths". Only octaves, fourths and fifths are truly harmonic intervals in this tuning. It is useful for solo parts, but not for conventional harmony, because of the poor major and minor thirds.

5.1.2 TEMPRMNT 2 (Just)

This tuning is unquestionably superior for unaccompanied vocal harmonic music. In instruments having a fixed intonation or in which retuning is a lengthy job (such as a harpsichord or piano), just intonation can only be used in keys which are harmonically closely related to the tonic. The tuning supplied is for the key of C.

5.1.3 TEMPRMNT 3 (Mean Tone)

This tuning is based upon perfect major and minor thirds, and is superior to both of the above tunings for accompanied harmonic music. Its name is derived from the major second interval which is the "mean" or average of the just intervals 10/9 and 9/8. Twelve tone versions of this tuning have a bad reputation because of the so-called "wolf-fifth" encountered in some keys. The more correct implementation of mean tone tuning has 27 keys to the octave, and the "wolf-fifth" does not arise. On keyboards of this sort, A-flat (for example) is a completely different note from G#. The 12-tone example given is attributed to Aron. It is best in C major, but works in keys requiring no more than 2 flats or 3 sharps.

5.1.4 TEMPRMNT 4 (Irregular)

The irregular temperament supplied is "Wirckmeister's Correct Temperment #3". The irregular temperaments (of which there were many) represent a transition between the truly harmonic tunings (such as 1 through 3 above) and the equal temperament used so widely today. They contain many intervals which are harmonically more pleasing than those of equal temperament, and yet have some transpositional flexibility. Keys harmonically distant from the tonic remain playable, but have subtle differences in mood which result from the greater prominence of the less harmonic intervals. These differences were exploited by Baroque composers and performers for aesthetic effect. The keyboard music of J.S. Bach is most properly played in a temperament of this sort, and in fact, Bach's title "Well Tempered Clavier" refers to such a temperament, not equal temperament as is presently widely believed.

5.2 USER SPECIFIED TOP OCTAVE

An alternate form of the TEMPRMNT statement allows user specification of the number of pitches per octave and the relative frequencies of each pitch in an octave. The statement itself specifies what is known as the "top octave" of note frequencies. The frequencies of notes in lower octaves are computed from the top octave frequencies by successive divison by 2.

The format of the alternate form is as follows:

```
TEMPRMNT N=n f1,f2,f3,f4,....,fn-1,fn
```

where n is the number of pitches per octave and f1, f2, ... are frequency parameters for each pitch in the top octave. The frequency parameters are actually the double-precision integer table increments used by the waveform scanning software for generating the notes of the top octave in the note table. For example, consider the statement:

```
TEMPRMNT N=12  17375, 17592, 18764, 19546, 20849, 23456, 25019, 26062,
               27365, 28147, 29319, 31274
```

The N=12 means that a user defined top octave of 12 tones is about to be defined, and that 12 numbers will follow. The first of these numbers, 17375,

specifies the lowest note of the top octave and is interpreted by the software as a double precision integer number representing the waveform table increment needed to produce 2326Hz which is C#7 in this tuning system. The last number, 31274, specifies the highest note of the top octave (4186Hz or C8 in this case). Detailed precedures for calculating waveform table increments are given in section 5.3.

The above example will form a 12-tone scale based upon the perfect harmonic intervals 10/9, 9/8, 6/5, 5/4, 4/3, 3/2, 8/5, 5/3, 7/4, 9/5, 15/8 and 2/1. It is not the same as the scale of just intonation loaded by the software when the TEMPRMNT 2 statement is used.

Note that the highest increment specified (the last one) need not be C8, nor does the number of tones have to be 12. However, if they are not, the note names (C, D, etc.) used in the Notes Section used to specify pitches will no longer specify the correct pitch. In this case the user is responsible for creating a note code translation table and he is responsible for the correct use of the OCTAVE statement. The OCTAVE statement will work with alternate note tables and it will correctly add or subtract a multiple of the number of tones in the octave, however, the system still allows only 98 notes in all. An offset (octave or half-step) which puts the interpreted note obtained from the note string out of this range will generate an error when the score is played. In the example given in section 5.4 for 19-tone equal temperament, a note name translation sheet is needed, having C6 as the top note (not C8 as before).

If the user attempts to specify a microtonal scale having more than 98 tones per octave, an error will result. In this case, the temperament software would be unable to completely load the top octave without overwriting part of the program.

Needless to say, music coded for a 12-tone scale is going to sound pretty weird if played on any other n-tone scale (and vice versa). However this will not generate an error and the music should still play unless notes go out of range as discussed above.

5.3 CALCULATION OF WAVEFORM TABLE INCREMENTS

The waveform table increments needed for a user specified top octave can be easily calculated from the pitches desired for the notes. The key fact is that the sample time for the digital synthesis part of the program is 114 microseconds. Initially, pitches are likely to be available in one of three forms:

1) Expressed as a frequency, in Hz, f.
2) Expressed as a ratio of whole numbers, r.
3) Expressed in "cents", c.

The first is easiest. If pitches are expressed in Hz, skip the following discussion. If not, the pitches must first be converted into frequencies. Cents are actually ratios expressed on a logarithmic basis. One cent is defined as 1/1200th of the logarithmic interval between 1 and 2. A frequency ratio between 1 and 2 can be obtained from a pitch value in cents by the following formula:

$$r = 2^{(c/1200)}$$

Alternatively, the pitches could have been expressed as whole number ratios, such as 3/2 for the Just Fifth Interval, or 5/4 for the Just Major Third, or as a decimal, 1.498307, for the 12-tone equal tempered fifth. To get a frequency

from a ratio we must have the frequency of some reference note. Usually the reference is C. The highest C which can be played by INSPLAY is C8 which has a frequency of 4186.01 Hz (this frequency was derived from the international pitch standard of 440.00000Hz for A4). This is used as the highest note of the top octave in all five of the temperaments and tunings supplied with INSMUS-8. Dividing the frequency of C8 by 2 gives the frequency of C7 (one octave below) and the basis for the top octave. If all the pitch ratios are multiplied by this value, the frequencies of the notes of the top octave will be obtained.

Having obtained the top octave note frequencies, the waveform table increment values must now be calculated. Since there are 256 entries (exactly one page of memory) in the waveform tables, and since the waveform scanning software can look up one of them and output it to the digital to analog converter in exactly 114 microseconds, a table increment of unity would work its way through a whole waveform table in:

$$t = .000114 * 256 = 0.029184 \text{ sec.}$$

This would generate a tone having a fundamental frequency of 1/t, or 34.265 Hz. In general, the frequency obtained is: f (Hz) = n/(.000114*256) = n/.029184 .

Conversely, the increment required to produce a given frequency is:

$$n = f\,(hz) * .000114 * 256$$

Thus for a frequency of 3027.5 hz, one would need a waveform table increment of 88.355. This decimal number and fraction must be converted into a double precision integer before it can be used as an argument for TEMPRMNT statement. This turns out to be simply a matter of multiplying the number by 256 again and rounding to the nearest integer value. Thus the overall formula for determining an argument for the TEMPRMNT statement from a top octave note frequency is:

$$N = f\,(Hz) * 7.471104$$

Thus N for a frequency of 3027.5Hz would be 3027.5 * 7.471104 = 22619.
Other examples are:

Pitch	R	Frequency based on C7	Wave table Increment	TEMPRMNT Argument
3/2	1.500000	3139.51	91.623	23456
5/4	1.250000	2616.26	76.353	19546
9/8	1.125000	2354.63	68.718	17592
100 cents	1.059463	2217.46	64.714	16567
200 cents	1.122462	2349.32	68.563	17552
800 cents	1.498307	3135.96	91.520	23429

5.4 A MICROTONAL EXAMPLE

As an illustrative example of a user specified temperament based on a scale which does not have 12 tones, we have chosen the 19-tone equal tempered scale. This scale has much to recommend it. It has scale intervals which are more highly consonant intervals than those of the 12-tone equal tempered scale we are all used to, and seven of the tones are quite close to the seven tones of the diatonic scale (actually closer than those of 12-tone equal temperament). This numerical coincidence should allow the INSMUS-8 synthesis software to play conventional music using 19-tone equal temperament (if a note code conversion is done which takes into account the key signature and the reduced pitch range of the note table in a 19-tone system).

Visualize a piano keyboard in which all of the black keys have been split to give two adjacent keys. That is, between F and G there are two additional keys, F# and G-flat, ditto for the other black keys. Furthermore, between E and F place a single key, shaped like a black key, but painted a different color (red for example). Place a similar key between B and C. This produces a keyboard having 7 white keys, 10 black keys, and 2 red keys, for each octave. Such a keyboard has been constructed and is quite playable. There are many possible consonant chords, some resembling the familiar major and minor chords, others totally new. For a detailed discussion of this keyboard and a statistical analysis of the degree of consonance of other equal tempered scales, see M. Yunik and G.W. Swift, "Tempered Music Scales for Sound Synthesis", page 60 of Vol. 4, No.4 of the Computer Music Journal, 1980. Unfortunately for the musician who would like to compose or perform music based upon the 19-tone equal tempered scale, you can't buy such a keyboard from the local music store.

The INSMSU-8 softwareare can be used to play music written for the 19-tone keyboard instrument described above, or indeed for any set of notes of well defined pitch. To do this, a note table of double precision table increments must be computed and written as a TEMPRMNT statement for the INSNOTRAN compiler. The TEMPRMNT statement needs the data as the "top octave" note increments, and calculates all the other note increments for lower octaves from them. The user must supply this data. But first he must figure out where on the frequency scale the top octave is.

The note frequency table that is ultimately created when the TEMPRMNT statement is executed has room for 98 note frequencies and silence. In the standard 12-tone tables which are supplied with INSMUS-8, the pitch range extends from the B below C0 up to C8. The default top octave thus contains C7# up to C8. The total range is 8 octaves plus 2 notes:

```
98/12 = 8 + 2/12
```

For a 19-tone note table we get:

```
98/19 = 5 + 3/19
```

This corresponds to 5 octaves plus 3 extra notes. This more limited range must be placed where it will be most useful. If we place the lowest note at B0-flat, then the highest note will be C6. This extends from 6 ledger lines below the bass clef up to 2 ledger lines above the treble clef. It is preferrable to have more lower notes available, since higher notes can easily be played using instruments containing higher harmonics (all even harmonics for example will double the apparent pitch).

At any one time only 63 pitches are actually available. To "reach" the other 35 pitches, it is necessary to use the OCTAVE (or OFFSET) statement. Note that the OCTAVE statement will indeed raise or lower the pitch by 19 tones after a TEMPRMNT statement with N=19 specified. Assuming use of OCTAVE, these 5+ octaves can be accessed in three ranges. For example:

```
-1 gives B0-flat up to D4-flat
 0 gives B1-flat up to D5-flat
+1 gives B2-flat up to C6
+2 is not valid
```

Note that in the third range, C6# and D6-flat are not valid. To make use of these ranges a typical OCTAVE statement might be:

```
OCTAVE V1=+1; V2=0; V3=0; V4=-1
```

This makes voice 1 a treble in the highest range, 2 and 3 are mid-range voices, and voice 4 is a bass.

The top octave in the above example is C5# to C6. These notes must be specified as waveform table increments, but first we need to know all the note frequencies. In equal tempered scales, the octave is divided up so that the frequency of each note is always a constant factor times the frequency of the next lower note. In the conventional scale this factor is the twelfth root of two: 2^(1/12)=1.059463. In a nineteen tone scale we need the nineteenth root of two: 2^(1/19)=1.037155.

Assuming the scale is to be based on C6 = 1047Hz, all the other notes are obtained from it by successive division by the factor 1.037155. We thus obtain, using the previous formulae:

Note Name	Frequency hz.	Table Increment	Note Name	Frequency hz	Table Increment
C6	1047.00	7822	F5#	726.96	5431
C6-flat	1009.49	7542	F5	700.92	5237
B5	973.33	7272	F5-flat	675.81	5049
B5-flat	938.46	7011	E5	651.60	4868
A5#	904.84	6760	E5-flat	628.25	4694
A5	872.43	6518	D5#	605.75	4526
A5-flat	841.17	6284	D5	584.05	4363
G5#	811.04	6059	D5-flat	563.12	4207
G5	781.98	5842	C5#	542.95	4056
G5-flat	753.97	5633	(C5	523.50	3911)

Having calculated the wave table increments for each note of the top octave, we now have enough information to complete the generalized **TEMPRMNT** statement:

```
TEMPRMNT N=19 7822, 7542, 7272, 7011, 6760, 6518, 6284, 6059, 5842, 5633,
              5431, 5237, 5049, 4868, 4694, 4526, 4326, 4207, 4056
```

Note that the increment for C5 was not used; it was included in the table above for example purposes only.

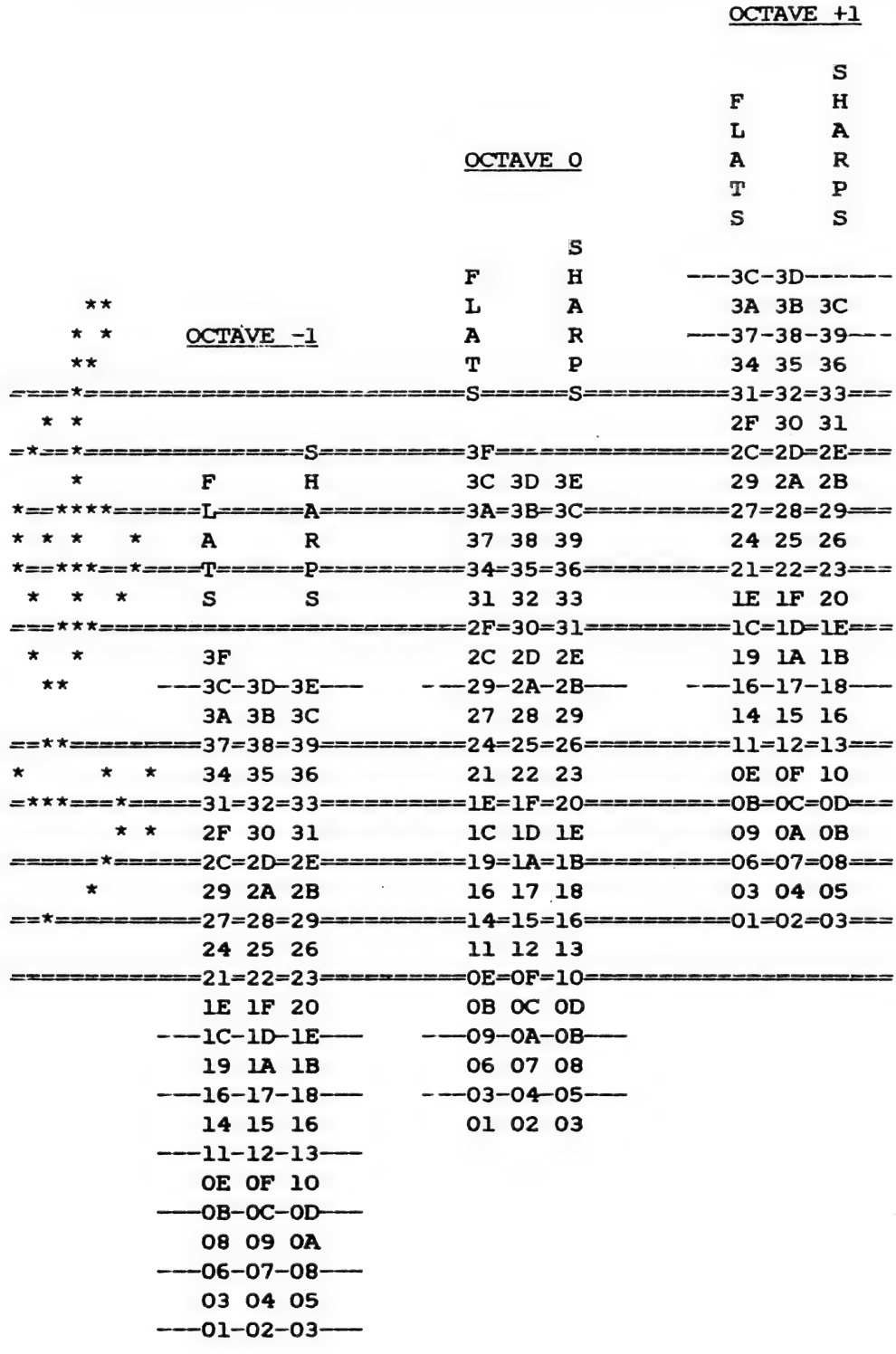
The above statement will indeed create the desired note frequency table. However, the note name compiler in INSNOTRAN will be nearly useless since it still outputs object code for a 12 tone scale. In this situation, it is best to revert to hexadecimal entry of notes in the Notes Section. In turn, it will be necessary to prepare a translation table that indicates what hex code to enter into the INSNOTRAN score to sound the desired 19 tone scale note. If there is a correspondence between the notes and those of traditional music, as there is in this case, a table based on the standard 12 tone musical staff is useful. The same translation table could be used for a 19-tone implementation of mean tone tuning, or indeed, any 19 tone scale.

Below is such a translation table that can be used to experiment with a 19 tone scale. To use it, find the physical position on the staff of the note you wish to sound and read off the hex equivalent to code in your score. Each group of 3 columns represents a different pitch range which can be selected with the OCTAVE statement. Within each group, the left column is for flats, the middle column is for naturals, and the right column is for sharps. Note that all the E# and F-flat notes have the same code, and that the codes for B# and C-flat are also the same. These notes are the "red" keys on the hypothetical keyboard. All the other sharps and flats have different pitches and are the doubled "black" keys. This table was prepared assuming that that only OCTAVE offsets

are used for accessing different pitch ranges.

The hex values in the table are all between 01 and 3F. To this must be added the playing mode for the note. Add hex 40 for an attack in mode 1, 80 for mode 2, and C0 for mode 3. Add nothing if the note is being continued from the previous event. It might be easiest to first code the notes in normal INSNOTRAN format, compile it, and then edit the hex object code in the listing into the format required for direct hex entry in the Notes Section. This way, you can let INSNOTRAN determine the event boundaries, modes, and attack/sustain information and you would only have to correct the actual pitch code for the 19 tone scale.

TRANSLATION TABLE FOR THE 19 TONE EQUAL TEMPERED SCALE



```
INSTRUMENT SYNTHESIS NOTRAN MUSIC COMPILER

--- COMMANDS SECTION ---

STMT LOCN OBJECT - BYTES  SOURCE STATEMENT
1                         * Example Score    Coded 8/3/82   G. Byrd
2                         *
3    2000 EE 00           LIB INSTFILE.I
4    2002 F0 04           NVOICES 4
5    2004                 LIBINS 1 W1 METSTRNG
                          * Pucked metallic stringed instrument used in
                          *     Sinfonia #14
                          * 18 Waveform Tables
                          * Optimum Tempo: 1/4=700
                          * Pitch Range: No low limit; high limit is D4
     2004 F5 02 13        WAVESET W1  N=18 H1 0,100; 16,0; H3 0,100; 14,0;
     2007                                  H5 0,100; 12,0; H7 0,75; 10,0;
     2007                                  H9 0,75; 8,0;   H11 0,68; 6,0;
     2007 01 00 3F 10 00                   H13 0,62; 4,0
     200C FF 03 00 3F 0E
     2011 00 FF 05 00 3F
     2016 0C 00 FF 07 00
     201B 2F 0A 00 FF 09
     2020 00 2F 08 00 FF
     2025 0B 00 2B 06 00
     202A FF 0D 00 27 04
     202F 00 FF 00
     2032 F6 01 00 02 13  SEQTAB 1  X=0 BR=00 W1
6    2037                 LIBINS 2 W2 GARGLE
                          * "Watery" instrument used in Sinfonia #14
                          * 17 Waveform Tables
                          * Optimum Tempo: 1/4=700
                          * Pitch Range: No low limit; high limit is C#5
     2037 F5 14 24        WAVESET W2  N=17 H2 0,0; 4,100; 16,0;
     203A                                  H3 0,0; 2,100; 5,0;
     203A                                  H4 4,0; 6,100; 13,0;
     203A                                  H5 0,0; 8,100; 16,0;
     203A                                  H6 5,0; 7,100; 9,0;
     203A 02 00 00 04 46                   H7 4,0; 9,100; 11,0
     203F 10 00 FF 03 00
     2044 00 02 46 05 00
     2049 FF 04 04 00 06
     204E 46 0D 00 FF 05
     2053 00 00 08 46 10
     2058 00 FF 06 05 00
     205D 07 46 09 00 FF
     2062 07 04 00 09 46
     2067 0B 00 FF 00
     206B F6 02 00 14 24  SEQTAB 2  X=0 BR=00 W2
7    2070                 LIBINS 3 W3 CATSTRNG
                          * Pucked catgut stringed instrument used in
                          *     Sinfonia #14
                          * 17 Waveform Tables
                          * Optimum Tempo: 1/4=700
                          * Pitch Range: No low limit; high limit is C#5
     2070 F5 25 35        WAVESET W3  N=17 H1 0,100; 16,0; H2 0,84; 14,0;
     2073                                  H3 0,68; 12,0;  H4 0,52; 10,0;
```

```
     2073                                    H5 0,36; 8,0;   H6 0,20; 6,0;
     2073 01 00 60 10 00                     H7 0,10; 4,0
     2078 FF 02 00 50 0E
     207D 00 FF 03 00 41
     2082 0C 00 FF 04 00
     2087 31 0A 00 FF 05
     208C 00 22 08 00 FF
     2091 06 00 13 06 00
     2096 FF 07 00 09 04
     209B 00 FF 00
     209E F6 03 00 25 35  SEQTAB 3  X=0 BR=00 W3
8    20A3 F7 43 08        MODIFY 4=3 1/32
9    20A6 F5 36 55        WAVESET W5 N=32 H1 0,0; 8,100; 31,0;
10   20A9                                 H2 0,0; 2,40; 31,20;
11   20A9                                 H3 0,0; 31,20;
12   20A9 01 00 00 08 BE                  H4 0,0; 4,60; 31,0; H5 0,0; 24,20
     20AE 1F 00 FF 02 00
     20B3 00 02 4C 1F 26
     20B8 FF 03 00 00 1F
     20BD 26 FF 04 00 00
     20C2 04 72 1F 00 FF
     20C7 05 00 00 18 26
     20CC FF 00
13   20CE F6 05 05 36 55  SEQTAB 5 X=0 BR=05 W5
14   20D3 E3 05           DRAWINS 5
15   20D5 F1 31 35 51     ASSIGN  M1 V1=3; V2=5; V3=5; V4=1
16   20D9 F1 32 41 11     ASSIGN  M2 V1=4
17   20DD F2 4F           TEMPO 1/4=600    * POINT A
18   20DF FE 0A 00        PLAY 10
19   20E2 FE 0A 00        PLAY 10
20   20E5 F3 95           OCTAVE V1=1
21   20E7 FE 14 00        PLAY 20
22   20EA F3 55           OCTAVE V1=0
23   20EC FE 1E 00        PLAY 30
24   20EF F2 56           TEMPO 3/12=650
25   20F1 FE 28 00        PLAY 40
26   20F4 F2 60           TEMPO 1/4=725
27   20F6 FE 32 00        PLAY 50
28   20F9 F2 6A           TEMPO 1/4=800
29   20FB FE 3C 00        PLAY 60
30   20FE F2 92           TEMPO 1/4=1100
31   2100 FE 46 00        PLAY 70
32   2103 FF              ENDCMD


--- NOTES SECTION ---

STMT LOCN OBJECT - BYTES  SOURCE STATEMENT
33                        ************************************************
34   2104                 MAXVOICE 4
35   2104                 KEY B@
36   2104                 SEGMENT 10
37   2104 30 66 00 00 5A   BE 1C4,1/8.;                          4C3,1/8.
38   2109 10 63 00 00 57      1A3,1/16;                         4A2,1/16
39   210E 40 5F 00 00 53      1F3,1/4;                          4F2,1/4
40   2113 40 63 00 00 53      1A3,1/4;                          4F2,1/4
41   2118 40 66 00 00 53      1C4,1/4;                          4F2,1/4
42   211D 80 6B 68 63 50      1F4,1/2;   2D4,1/2;   3A3,1/2;   4D2,1/2
43   2122 30 6F 6A 63 4F      1A4,1/8.;  2E4,1/8.;  3A3,1/8.;  4C#2,1/8.
44   2127 10 6D 00 61 4F      1G4,1/16;             3G3,1/16;  4C#2,1/16
```

```
45   212C 40 6B 00 5F 50      1F4,1/4;                3F3,1/4;    4D2,1/4
46   2131 40 63 00 5F 50      1A3,1/4;                3F3,1/4;    4D2,1/4
47   2136 40 65 61 5E 55      1B"3,1/4;  2G3,1/4;     3E3,1/4;    4G2,1/4
48   213B 80 66 61 5E 5A      1C4,1/2;   2G3,1/2;     3E3,1/2;    4C3,1/2
49   2140 20 66 00 00 5A      1C4,1/8;                            4C3,1/8
50   2145 20 66 00 00 5A      1C4,1/8;                            4C3,1/8
51   214A 60 6F 6B 66 53   C  1A4,1/4.;  2F4,1/4.;   3C4,1/4.;   4F2,1/4.
52   214F 20 6D 00 61 55      1G4,1/8;                3G3,1/8;    4G2,1/8
53   2154 40 6B 00 5F 57      1F4,1/4;                3F3,1/4;    4A2,1/4
54   2159 80 6A 66 61 5A      1E4,1/2;   2C4,1/2;     3G3,1/2;    4C3,1/2
55   215E 30 68 64 00 58      1D4,1/8.;  2B3,1/8.;                4B2,1/8.
56   2163 10 6A 66 61 58      1E4,1/16;  2C4,1/16;    3G3,1/16;   4B2,1/16
57   2168 40 6B 66 63 57      1F4,1/4;   2C4,1/4;     3A3,1/4;    4A2,1/4
58   216D 40 6B 00 00 5F      1F4,1/4;                            4F3,1/4
59   2172 40 66 00 00 5A      1C4,1/4;                            4C3,1/4
60   2177 40 63 00 00 57      1A3,1/4;                            4A2,1/4
61   217C 40 5F 00 00 53   DE 1F3,1/4;                           4F2,1/4
62   2181 00                  ENDSEG
63   2182                     SEGMENT 20
64   2182 30 AF 6B 63 53      1A4,1/8.M2;2F4,1/8.;    3A3,1/8.;   4F2,1/8.
65   2187 10 6F 6B 63 53      1A4,1/16M1;2F4,1/16;    3A3,1/16;   4F2,1/16
66   218C 40 6F 6B 63 53      1A4,1/4;   2F4,1/4;     3A3,1/4;    4F2,1/4
67   2191 40 70 6B 64 55      1B4,1/4;   2F4,1/4;     3B3,1/4;    4G2,1/4
68   2196 40 72 6B 66 57      1C5,1/4;   2F4,1/4;     3C4,1/4;    4A2,1/4
69   219B 80 B2 6B 66 57      1C5,1/2M2; 2F4,1/2;     3C4,1/2;    4A2,1/2
70   21A0 20 70 6B 64 55      1B4,1/8M1; 2F4,1/8;     3B3,1/8;    4G2,1/8
71   21A5 20 6F 6B 63 53      1A4,1/8;   2F4,1/8;     3A3,1/8;    4F2,1/8
72   21AA 40 6D 6A 61 5A      1G4,1/4;   2E4,1/4;     3G3,1/4;    4C3,1/4
73   21AF 40 6F 6B 63 5F      1A4,1/4;   2F4,1/4;     3A3,1/4;    4F3,1/4
74   21B4 40 70 6A 64 61      1B4,1/4;   2E4,1/4;     3B3,1/4;    4G3,1/4
75   21B9 80 B0 6D 6A 5A      1B4,1/2M2; 2G4,1/2;     3E4,1/2;    4C3,1/2
76   21BE 40 70 6A 64 4E      1B4,1/4M1; 2E4,1/4;     3B3,1/4;    4C2,1/4
77   21C3 60 AF 66 63 53      1A4,1/4.M2;2C4,1/4.;    3A3,1/4.;   4F2,1/4.
78   21C8 20 6D 66 61 55      1G4,1/8M1; 2C4,1/8;     3G3,1/8;    4G2,1/8
79   21CD 40 6B 66 5F 57      1F4,1/4;   2C4,1/4;     3F3,1/4;    4A2,1/4
80   21D2 80 AA 66 61 5A      1E4,1/2M2; 2C4,1/2;     3G3,1/2;    4C3,1/2
81   21D7 20 68 00 5C 58      1D4,1/8M1;              3D3,1/8;    4B2,1/8
82   21DC 20 6A 66 5E 58      1E4,1/8;   2C4,1/8;     3E3,1/8;    4B2,1/8
83   21E1 40 6B 66 5F 57      1F4,1/4;   2C4,1/4;     3F3,1/4;    4A2,1/4
84   21E6 40 63 5F 00 5C      1A3,1/4;   2F3,1/4;                 4D3,1/4
85   21EB 40 65 61 5F 53      1B"3,1/4;  2G3,1/4;     3F3,1/4;    4F2,1/4
86   21F0 80 A6 61 5E 4E      1C4,1/2M2; 2G3,1/2;     3E3,1/2;    4C2,1/2
87   21F5 00                  ENDSEG
88                            * G
89   21F6                     SEGMENT 30
90   21F6 40 66 00 00 5A      1C4,1/4M1;                          4C3,1/4
91   21FB 40 6B 66 63 53      1F4,1/4;   2C4,1/4;     3A3,1/4;    4F2,1/4
92   2200 40 6B 64 61 55      1F4,1/4;   2B3,1/4;     3G3,1/4;    4G2,1/4
93   2205 20 6B 66 63 57   H  1F4,1/8;   2C4,1/4;     3A3,1/4;    4A2,1/4
94   220A 20 6A 26 23 17      1E4,1/8
95   220F 40 68 64 5F 58      1D4,1/4;   2B3,1/4;     3F3,1/4;    4B2,1/4
96   2214 40 68 64 5F 58      1D4,1/4;   2B3,1/4;     3F3,1/4;    4B2,1/4
97   2219 40 68 66 60 57      1D4,1/4;   2C4,1/4;     3F#3,1/4;   4A2,1/4
98   221E 40 6D 64 61 55      1G4,1/4;   2B3,1/4;     3G3,1/4;    4G2,1/4
99   2223 20 70 64 00 55      1B4,1/8;   2B3,1/8;                 4G2,1/8
100  2228 20 6F 63 00 57      1A4,1/8;   2A3,1/8;                 4A2,1/8
101  222D 20 6D 61 00 58      1G4,1/8;   2G3,1/8;                 4B2,1/8
102  2232 20 6B 5F 00 58      1F4,1/8;   2F3,1/8;                 4B2,1/8
103  2237 40 6B 66 63 5A      1F4,1/4;   2C4,1/4;     3A3,1/4;    4C3,1/4
104  223C 40 6A 66 61 4E      1E4,1/4;   2C4,1/4;     3G3,1/4;    4C2,1/4
```

```
105  2241 20 66 00 00 5A      1C4,1/8;                          4C3,1/8
106  2246 20 66 00 00 58      1C4,1/8;                          4B2,1/8
107  224B 60 6B 66 63 57      1F4,1/4.;  2C4,1/4.;  3A3,1/4.;   4A2,1/4.
108  2250 00                 ENDSEG
109  2251                    SEGMENT 40
110  2251 20 6D 00 61 5A   I  1G4,1/8;              3G3,1/8;    4C3,1/8
111  2256 20 6F 00 63 5F      1A4,1/8;              3A3,1/8;    4F3,1/8
112  225B 20 70 00 64 61      1B4,1/8;              3B3,1/8;    4G3,1/8
113  2260 00                 ENDSEG
114  2261                    SEGMENT 50
115  2261 80 72 6F 6B 63   J  1C5,1/2;   2A4,1/2;   3F4,1/2;   4A3,1/2
116  2266 20 6B 68 63 5C      1F4,1/8;   2D4,1/8;   3A3,1/8;   4D3,1/8
117  226B 20 6D 65 5F 5C      1G4,1/8;   2B~3,1/8;  3F3,1/8;   4D3,1/8
118  2270 60 6F 66 63 5A      1A4,1/4.;  2C4,1/4.;  3A3,1/4.;  4C3,1/4.
119  2275 00                 ENDSEG
120  2276                    SEGMENT 60
121  2276 20 70 6B 68 5A   K  1B4,1/8;   2F4,1/8;   3D4,1/8;   4C3,1/8
122  227B 48 6D 6A 64 5A      +1G4,1/4;   2E4,1/4;   3B3,1/4;   4C3,1/4
123  2280 00                 ENDSEG
124  2281                    SEGMENT 70
125  2281 A2 6B 66 63 53      ++1F4,1/2;  2C4,1/2;  3A3,1/2;   4F2,1/2
126  2286 00                 ENDSEG
127                          *
128  2287 00                 END
```

A number of error conditions are possible while using the INSNOTRAN compiler. Listed below is each error that may be detected, system action in response to the error condition, and possible corrective action the user may take.

Initial User Dialog Errors

These errors may occur when giving the file names at the beginning of an INSNOTRAN run. Each allows re-entry of the erroneous information. If you wish to abort the compilation instead, hold the CTRL key down and type a C which will give the CODOS> prompt.

DEVICE NAME NOT ALLOWED, TRY AGAIN. - When specifying the source or object file name you must specify a disk file because the compiler may have to "backup" in the file during the compilation process. Use a system utility to transfer the source statements from the device to a disk file and then run the compiler again.

ILLEGAL FILE OR DEVICE NAME, TRY AGAIN. - The file or device name given does not conform to the CODOS rules for such names. Check the spelling or refer to section 2 of the CODOS manual.

SPECIFIED DISK DRIVE NOT OPEN, TRY AGAIN. - The file name specified is on a disk drive which does not have a diskette inserted into it or which has not been opened with an OPEN command. Check the file name spelling or insert the needed diskette and OPEN the drive.

SPECIFIED FILE DOES NOT EXIST, TRY AGAIN. - The name given for the source file name cannot be found. Check the spelling and drive number.

SPECIFIED DISK DRIVE IS WRITE PROTECTED, TRY AGAIN. - The disk drive specified in the listing or object file name is write protected. Specify a file on a different drive or exit the compiler, affix a write permit sticker to the disk, and rerun the compiler.

SPECIFIED FILE IS LOCKED, TRY AGAIN. - An existing file name has been given for either the listing or the object file but it is write protected. Either give another file name or exit the compiler, UNLOCK the file name, and then re-run the compiler.

SPECIFIED FILE ALREADY EXISTS, OK TO OVERWRITE? - An existing file name has been given for either the listing or the object file. Type a Y if you wish this file to be replaced with the results of this run. Type an N if you would like to specify a different file name.

Compilation Errors Printed on the Listing

These errors relate directly to the statements in the INSNOTRAN source file. If there is more that one error indication on a single line, pay special attention to the first one printed since the condition that caused it may have spilled over to cause the others.

NO END STATEMENT - The end of the source file has been reached without encountering an END statement. Use the editor to add the needed END statement and compile again.

UNDEFINED SEGMENT ID - XXXX - A PLAY statement in the Commands Section called for a segment ID (XXXX) that was never seen in a SEGMENT statement in the Notes Section. The address field of the compiled PLAY statement is <u>not</u> updated therefore the music will go wild when it reaches this PLAY statement.

ER 1-INVALID KEYWORD - A keyword has been misspelled. The statement is ignored. Use the editor to correct the spelling and compile again.

ER 2-INVALID NUMBER - A digit or number could not be found where expected. The remainder of the statement (except NEWINS and WAVESET) is ignored. Check the statement syntax or the keyword spelling.

ER 3-INVALID DELIMITER - The character immediately following a digit or number is not legal. The remainder of the statement (except NEWINS, WAVESET and note statements) is ignored. Remember that blanks may not be embedded <u>within</u> note specifications.

ER 4-NUMBER IS OUT OF RANGE - A number in the statement is either too small or too large. The statement is compiled with a default value substituted for the out of range value. Check the limitations that apply to the statement and correct.

ER 5-INVALID TEMPO FRACTION - Either the numerator or denominator is greater than 255 or there are embedded blanks or the / is omitted. The statement is ignored.

ER 6-INVALID TEMPO DURATION - The number following the equals has an illegal character or the = after the tempo fraction was not found. The statement is ignored. Remember that commas must not be embedeed in the duration number.

ER 7-TEMPO TOO SLOW - The tempo specified is slower than 7.4 seconds for a whole note. Use a faster tempo and double the durations of the notes. This can also be caused by overflow in the tempo calculation. Reduce the tempo fraction to lowest terms and try again.

ER 8-TEMPO TOO FAST - The tempo specified is so fast that all significance is lost in the internally calculated value. This is usually caused by overflow in the tempo calculation. Reduce the tempo fraction to lowest terms and try again.

ER 9-ILLEGAL WAVEFORM SET NUMBER - The ID number used to refer to a waveform set is zero or greater than 16 or has some other kind of error. The statement containing the error is ignored.

ER 10-ILLEGAL OVERALL AMPLITUDE - The overall amplitude in a NEWINS or WAVESET statement is greater than 100 or has some other kind of error. The statement is ignored.

ER 11-ILLEGAL HARMONIC NUMBER - The harmonic number is zero or has some other kind of error. The harmonic specification is ignored.

ER 12-ILLEGAL HARMONIC AMPLITUDE - The harmonic amplitude is greater than 100 or has some other kind of error. An amplitude of zero is substituted.

ER 13-ILLEGAL WAVEFORM TABLE NUMBER - The waveform table number specified in a harmonic specification of a WAVESET statement is equal to or greater than the number of tables (N parameter). N-1 is substituted.

ER 14-ILLEGAL NOTES SEGMENT ID - The ID number of the SEGMENT statement has a syntax error such as an embedded comma. The statement is ignored.

ER 15-NO NOTES SECTION BEFORE END - You have probably forgotten to place an ENDCMD statement at the end of the Commands Section. The compiler terminates normally but the object file will not be usable.

ER 16-INVALID SEGMENT ID - Identical to ER 14.

ER 17-DUPLICATE SEGMENT ID - The ID number specified in the SEGMENT statement has already been used in a prior SEGMENT statement. The statement is ignored.

ER 18-WARNING - NOTES STILL SOUNDING AT END OF SEGMENT - When an ENDSEG statement is encountered, there were still some notes sounding from previous statements. The most likely cause is an error in the duration field of some prior note statement. The condition is ignored. When the piece is played, the offending notes will be truncated at the segment boundary.

ER 19-INVALID KEYLETTER IN NOTE STATEMENT - A line with a blank or digit in column 1 was encountered but a note or a rest specification could not be found. Most likely cause is a forgotten * in a comment statement or not starting the note or rest specifications in column 5 or beyond. The statement is ignored.

ER 20-INVALID CHARACTER IN REST SPECIFICATION - Most likely a comma is missing between the R and the duration fraction. The rest specification is skipped.

ER 21-INVALID DURATION SPECIFICATION - Any kind of error in the duration fraction and following dots, if any. Remember that the final evaluated duration including dots must not be greater than a whole note.

ER 22-VOICE NUMBER OUT OF RANGE - In a note specification, the specified voice number is zero or greater than 4 or has a syntax error. The note specification is skipped.

ER 23-ILLEGAL PITCH SPECIFICATION - The pitch letter is not A-G or is not upper case or the final evaluated pitch including sharps and flats is below C1 or above C6 or there is some other kind of syntax error. The note specification is ignored but any others on the line are processed.

ER 24-INVALID CHARACTER IN NOTE SPECIFICATION - An illegal modifier character was seen after the duration specification. The note specification is ignored.

ER 25-VOICE NUMBER GREATER THAN CURRENT MAXVOICE - Self explanatory.

ER 26-VOICE STILL SOUNDING FROM PREVIOUS LINE(S) - A voice has been instructed to play a note before one started in a previous statement has finished. The most likely cause is an error in the duration in a previous note statement. The previous note is truncated and the specified one played.

ER 27-ENDSEG WITHOUT MATCHING SEGMENT - An ENDSEG statement was encountered without previously having seen a SEGMENT statement. The statement is ignored.

ER 28-MAXVOICE CHANGE INSIDE A SEGMENT - The maximum number of voices may only be changed between segments. The statement is ignored.

ER 29-NOTES ENCOUNTERED OUTSIDE OF A SEGMENT* - There was no SEGMENT statement before the note statements began. Can also be caused by not starting a command in column 1 in the Notes Section. The notes are compiled but will not be played since they cannot be addressed by a PLAY statement.

ER 30-ONE OR MORE PARAMETERS MISSING* - In an ASSIGN statement, not all voices were assigned. The missing parameters are assumed to be zero.

ER 31-VOICE SPECIFIED MORE THAN ONCE* - In a note statement, a voice is playing more than 1 note. The first note encountered for that voice is played.

ER 32-NOISE SPECIFIED BEFORE LAST HARMONIC* - In a NEWINS or a WAVESET statement, a harmonic number less than 128 was specified after a noise "harmonic" was specified. Re-order the harmonic specifications.

ER 33-TABLE NUMBER OR TIME NOT IN ASCENDING ORDER* - In a NEWINS or a WAVESET statement, the X coordinates of the line segments are not specified in ascending sequence. The most likely cause is a typing error.

ER 34-OUT OF MEMORY -- COMPILATION ABORTED* - The INSNOTRAN compiler has determined that there will be insufficient memory to hold the score and the generated waveform tables when the score is played. You will have to either reduce the number of waveform tables used by instrument definitions, reduce the number of instrument definitions, or reduce the size of the score.

ER 35-TOO MANY HARMONIC SPECIFICATIONS* - The amplitude optimization routine used in compiling NEWINS or WAVESET statements has found more than approximately 55 non-zero harmonic specifications and has insufficient memory available to perform. The remaining harmonic specifications are ignored.

ER 36-TIME PARAMETER OUT OF RANGE* - The time value specified in a harmonic specification of a NEWINS statement is greater than the overall envelope duration which is a whole note duration at the specified average tempo. Most likely due to a typing error.

ER 37-DEVICE NOT ALLOWED AS LIBRARY* - The name specified for a library file in a LIB statement must be the name of a disk file.

ER 38-ILLEGAL NAME/DRIVE NUMBER* - The name specified for a library file in a LIB statement is illegal or specifies an illegal disk drive number.

ER 39-DRIVE NOT OPEN* - The drive number specified for a library file is not open. Open the drive or correct the LIB statement and recompile.

ER 40-FILE NOT FOUND* - The name specified for a library file cannot be found. Insert the correct disk or correct the LIB statement and recompile.

ER 41-LIBRARY CHANNEL NOT ASSIGNED* - A LIB statement did not appear before a LIBINS statement was encountered.

ER 42-INSTRUMENT NOT FOUND* - The specified instrument name was not found in the library file specified by the previous LIB statement.

ER 43-ILLEGAL INSTRUMENT ID* - Any place an instrument ID is required, the ID number is less than 1 or greater than 15.

ER 44-LARGER THAN ORIGINAL DEFINITION -- CANNOT OVERWRITE - In a WAVESET or NEWINS statement that specifies the same waveset number as an earlier WAVESET or NEWINS statement, more waveforms were specified than what the earlier statement specified and thus cannot fit into the memory space already allocated. If you actually wish to overlap waveform sets like this, either specify a larger N for the earlier statement or reorder the statements.

ER 45-ILLEGAL STEREO ASSIGNMENT - The LEFTCHAN statement specified an odd number of voices to be in the left channel.

ER 46-ILLEGAL LIBRARY COMMAND - There is a syntax error of some kind in a LIBINS statement.

ER 47-EXCESSIVE PITCH OFFSET - The pitch offset specified or the cumulative total of relative offsets in an OFFSET statement is less than -12 or greater than +24 semitones. The octave offset specified in an OCTAVE statement is less than -1 or greater than +2 octaves.

ER 48-WAVEFORM SET NOT DEFINED - A SEQTAB or QSEQTAB statement specified a waveform table set that had not been previously created.

ER 49-FORWARD STEP MUST BE GREATER THAN BACKWARD STEP - In a WARBLE statement, the FS parameter is less than the BS parameter thus causing a net backward movement through the tables.

ER 50-NUMBER TO BIG - This is a general catchall error for numerical parameters that are too large.

INSPLAY checks for several types of error conditions while it is playing a score. Many of these would normally be caught while compiling the score but if use is made of QWAVESET, QSEQTAB, or direct hex entry, these conditions could be encountered. The error code is displayed by INSPLAY in the log area and in most cases will cause an abort of the offending song.

INVALID # OF VOICES - The number of voices specified by a compiled NVOICES statement is not between 1 and 4 inclusive.

INVALID STEREO PATTERN - The stereo pattern specified by a compiled LEFTCHAN statement has an odd number of voices in one of the channels.

INVALID MODE - The playing mode specified in a compiled ASSIGN statement is not 1, 2, or 3.

INVALID PITCH OFFSET - The pitch offset specified by a compiled OCTAVE or OFFSET statement exceeds the range of -12 to +24 semitones.

PAGE REVERSAL OR OUT OF RANGE - The sequence of waveform table numbers specified by a compiled NEWINS or WAVESET statement for linear interpolation of harmonic envelopes is not strictly ascending.

PREMATURE NOISE CALL - In a compiled NEWINS or WAVESET statement, a harmonic number less than 128 follows a harmonic number greater than 128.

WAVEFORM OVERFLOW - When computing a waveform set, an overflow occured. Most likely the result of a typing error in a QWAVESET statement.

ATTEMPT TO OVERWRITE COMMAND ORIGIN - The memory area for waveform sequence tables has extended down below the beginning of the score thus wiping it out. Most likely the result of incorrect patching of the object code.

INVALID I.D. - An invalid instrument ID was encountered in a compiled SEQTAB, QSEQTAB, WARBLE, or MODIFY statement. The ID must be between 1 and 15 inclusive.

OVERSPECIFICATION - In a compiled QSEQTAB statement, the waveform sequence table was filled but data still remains. Most likely the result of a typing error in the QSEQTAB statement.

INVALID STEP SPECIFICATION - An error in either the FS or BS specification in a compiled WARBLE statement.

BACKWRD STEP > FRWD STEP - In a compiled WARBLE statement, the BS parameter is greater than the FS parameter.

INVALID COMMAND - An undefined op code was encountered while interpreting the compiled score. Most likely due to trying to play a file which is not a compiled INSNOTRAN object code file.

HILIM = OR < THAN LOWLIM - An error in a set LOWLIM instruction caused the memory space allocated for waveform tables to become zero. Most likely due to trying to play a file which is not a compiled INSNOTRAN object code file.

UNDEFINED INSTRUMENT - A compiled ASSIGN statement specified an instrument ID that had not yet been defined.

INVALID TEMPERAMENT PARAMETER - A compiled TEMPRMNT statement specified a temperament number greater than 4.

USER ABORT - The user pressed the f1 function key while the song was being played.

Throughout the preceeding sections, various cautions and system limitations have been mentioned. All of this important data has been combined to make up this section. In many cases, exceeding a limitation does not produce an immediately harmful effect, it merely produces an unexpected or ill-behaved effect. Indeed, many special effects can be produced by experimenting with illogical or illegal combinations of specifications.

1. ALIASING - Due to the sampled data reconstruction of waveforms utilized, unexpected results can occur when trying to synthesize frequencies above 1/2 the 8.77KHz sample rate which is used in this version. The rule is that the highest harmonic of the highest note played with an instrument specification must be less than 3.5KHz. For example, if C5 (522Hz) is the highest note played by voice 2, then the instrument played by voice 2 should not use harmonics higher than the sixth. Honoring this may require that a different instrument specification be used for the very high notes. See section 4.5.1.

2. WAVEFORM SEQUENCE TABLE WRAPAROUND - If a note is continuously sustained longer than 256 "tempo units" the waveform sequence table will "wraparound" and start a new attack. This will occur with anything longer than a whole note when using INSNOTRAN to compile scores. The problem may be overcome by temporarily halving the tempo and then halving the duration specifications of notes to compensate.

3. OVERFLOW IN WAVEFORM CALCULATIONS - This is only possible when the QWAVESET statement is used. Overflow is evidenced by a loud raspy buzz or static sound that completely dominates the instrument. The cause is probably an error in preparing the QWAVESET statement from a previously compiled WAVESET or NEWINS statement.

4. CONSTANT ENVELOPE DURATIONS - The audible duration of envelopes is determined by the waveform sequence table used in the instrument definition, not the event durations in the note string. While not a limitation for rapidly decaying plucked style instruments, it can become a problem with wind or other sustained instruments. The best way to overcome this problem is to construct several waveform sequence tables (instruments) with the sustain portion set to different lengths and then use the one most appropriate for the note duration. If three variations are sufficient, one may then specify the envelope by using the modes feature. The only limitation to this technique is the maximum of 15 instruments in memory.

5. CLICKS BETWEEN EVENTS - Since the same microprocessor is used to generate sound and to set up parameters for an event, sound generation necessarily stops between events. The resulting click is minimized as much as possible by interrupting only at waveform zero crossings and by optimizing the parameter setup process. The effect is much less noticable when several instruments are playing at once with complex waveforms than when one instrument is playing with a simple waveform. In solo or duet cases, some improvement is possible by reducing the number of voices with an NVOICES statement.

6. While not a limitation it is helpful to know that waveform tables use offset binary coding for positive and negative numbers. In offset binary, hex 80 is equivalent to zero, hex FF is equivalent to +127, and hex 00 is equivalent to -128. Likewise, an offset binary coded audio DAC is expected.

INSNOTRAN

Address Range	Description
0000 - 00A9	Page zero temporary storage
0700	INSNOTRAN entry point
0700 - 31B9	Program code
31BA - 321F	Pointers to error messages
3220 - 39CA	Error message text
39D5 - 3A15	1/4 cycle sine table, unsigned values
3B00 - 3BFF	Input line buffer
3C00 - 3CFF	Output line buffer
3D00 - 44FF	List of locations in Commands Section code to fix up
4500 - 4CFF	List of segment ID numbers in Notes Section
4D00 - 54FF	List of corresponding segment addresses in Notes Section
5500 - 58FF	Buffer for source file
5900 - 5CFF	Buffer for listing file
5D00 - 60FF	Buffer for library file
6100 - 61FF	Buffer for temporary object code
6200 - 67FF	Miscellaneous storage for NEWINS and WAVESET optimization
6800 - BBFF	Amplitude table storage for NEWINS and WAVESET optimization

INSPLAY

Address Range	Description
0000 - 00FF	Page zero storage (saved while running, restored when done)
0700	INSPLAY Entry point
0738 - 0746	Configuration string
0747 - 1E9D	Main program code
1EF4 - 1F34	1/4 sine wave table, unsigned values
1F35 - 1FBC	Additional storage
2000 - B8FF	Storage for score, sequence tables, and waveform tables
B900 - B9FF	Copy of page zero
BA00 - BA3D	OP code jump table
BA3E - BAB5	Temperament top octave frequency tables
BAB6 - BADD	Miscellaneous storage
BADE - BBA3	Generated complete note frequency table
BBA4 - BBEC	Miscellaneous storage
BD00 - BDFF	Input line buffer for edit functions

Because of the very technical nature of digital sound synthesis, this manual of necessity uses some terms that may not be familiar to the average computer user. The following glossary is not meant to be exhaustive but does list those terms that may be difficult to find in standard dictionaires of computer related terms.

ALIAS - The incorrect frequency assumed by a frequency component higher than 1/2 of the sample frequency in a waveform sampling system.

AMPLITUDE - The magnitude of vibration of a sound waveform; exponentially related to perceived loudness.

ATTACK - The build-up of sound loudness at the beginning of a note.

CHIFF - A puffing sound during the attack of organ pipe tones. Although organ designers try to minimize it, it is normally considered to be the hallmark of "accurate" organ tone synthesis.

COMMANDS SECTION - The first half of an INSNOTRAN score that directs the computation of instrument waveforms, the setting of synthesis parameters, the tempo and other "style" variables, and the playing of segments of the song.

DAC - Acronym for Digital-to-analog converter which is the device used to convert the digital representation of sound to an audio signal.

DEFAULT - A parameter setting or definition that is used in the absence of any contrary setting in the score.

DURATION - The length of time that a musical note sounds. It is usually specified as a fraction of a measure.

DURATION BYTE - The first byte in the object code for an event in the song string which specifies the time length of the event in terms of tempo units.

DYNAMIC TIMBRE VARIATION - The change in harmonic composition of a note during its duration.

ENVELOPE - The curve that represents the loudness variation during a note of either the entire tone or one of its harmonics.

EVENT - A period of time during which the chord structure of the overall musical sound is constant.

FREQUENCY - The number of times per second that a sound waveform repeats. Exponentially related to musical pitch.

FUNDAMENTAL - A component of a sound wave having the same frequency as the overall wave.

HARMONIC - A component of a sound wave having a frequency that is an integral multiple of the frequency of the overall wave.

INSTRUMENT - The combination of a waveform set and waveform sequence table that represents a musical instrument sound. Instruments are numbered from 1 to 15.

MODE - An indirect instrument selection (out of 3 possible) coded directly in the pitch bytes in the note string.

MUSIC INTERPRETER - The INSPLAY program in this package that interprets the object code produced by the INSNOTRAN compiler and performs waveform computation and sound generation.

NOISE COMPONENT - The random component in conventional musical instrument sound waveforms.

NOTE STRING - A sequence of object code bytes that specifies the actual notes to be played in a musical composition.

NOTES SECTION - The second half of an INSNOTRAN score which contains the actual notes of the song divided into one or more segments.

OBJECT - The binary machine reabable representation (i.e., compiled) of a program or INSNOTRAN musical score.

OCTAVE OFFSET - A parameter that directs the music interpreter to sound notes in different octaves than coded in the note string.

PITCH - The degree of perceived highness or lowness of a tone. Logarithmically related to frequency.

PITCH BYTE - A component of the note string object code that specifies the musical pitch to be sounded by a voice during an event.

PITCH OFFSET - A parameter that directs the music interpreter to sound notes in a different musical key than coded in the note string.

RELEASE - The decay of sound loudness at the end of a note.

REST - That period of time during which a voice generates no sound.

SAMPLE PERIOD - The time between sound wave samples presented to the digital-to-analog converter. The sample period in INSMUS-8 is 114 microseconds.

SAMPLE RATE - The frequency at which sound wave samples are presented to the digital-to-analog converter. The sample rate in INSMUS-8 is 8772Hz.

SEGMENT - A portion of a musical score that is treated as a unit and played with constant tempo, instrument assignments, and other overall system parameters. Each segment in the score is given an arbitrary identification number between 1 and 65535.

SILENT WAVEFORM - The waveform table that is always present as instrument 0 which is filled with audio zero values (hex 80's).

SEQUENCE TABLE - A list of 256 bytes that refer to the members of a waveform set which defines the sound of an instrument.

SOURCE - The human readable textural representation (i.e., uncompiled) of a program in a programming language or a score in INSNOTRAN.

STEREO PATTERN - The correspondance between voices and their location in stereo space.

SUSTAIN - The time in the middle of a note where the loudness is relatively steady.

TEMPO - The speed at which a musical composition is played. The numerical value of the Tempo parameter (larger values give slower speeds).

TEMPO PERIOD - A period of time equal to the product of the sample period (114 microseconds) and the Tempo parameter.

TIMBRE - The "tone color" of a musical note. Influenced by its harmonic composition and envelope.

VOICE - A musical line or sound source that is independent of other sound sources. INSMUS-8 has a maximum of 4 voices.

WAVEFORM SEQUENCE TABLE - See sequence table.

WAVEFORM SET - A set of related, stored waveform tables. Each waveform in the set uses 1/4K of memory (256 bytes).

WAVEFORM TABLE - A list of 256 numbers between -128 and +127 that represents exactly one cycle of a sound waveform in digital form. Best visualized as a circular table where the end joins the beginning.

WAVEFORM TABLE INCREMENT - A number that determines how far the waveform table scanning pointer should advance each sample period. The larger the number, the more rapidly the pointer moves around the waveform table and the higher the waveform frequency.

The field of computer synthesized music is an exciting and dynamic one with almost unlimited oppertunities for individuals to make significant contributions using personal computers such as the MTU-130/140. The following publications are suggested for further reading and study:

BOOKS

1. Chamberlin. H. A. Musical Applications of Microprocessors. Rochelle Park, New Jersey: Hayden Book Co., 1980. (available from MTU, K-1002-BOOK).

2. Mathews, Max V. The Technology of Computer Music. Cambridge, Massachusetts: MIT Press, 1969.

3. Morgan, C. P. The Byte Book of Computer Music. Peterborough, New Hampshire: Byte Publications, 1979.

4. Proceedings, Symposium on Small Comouters in the Arts. IEEE Catalog No. 81CH1721-0, 1981, 1982, 1983.

PERIODICALS

1. Electronotes Newsletter, 1 Pheasant Lane, Ithaca, New York 14850.

2. Computer Music Journal, MIT Press, Cambridge, Massachusetts.

ARTICLES

1. Chamberlin, H. A. "A Sampling of Techniques for Computer Performance of Music" Byte Magazine, September, 1977. (Available from MTU, K-1002-1ART).

2. Chamberlin, H. A. "Advanced Real-Time Music Synthesis Techniques" Byte Magazine, April, 1980 (Available from MTU, K-1002-6ART, contains a description of the inner sound generation loop of INSPLAY).

For stereo operation, you should purchase a K-1002 8-bit Audio System board from MTU or an equivalent and connect it to port A on the computer's parallel I/O connector. See the Hardware Documentation section of the MTU-130/140 manual for pin connections on the parallel port. The +5 volt operating power needed by the board is also available on this connector.

Then, assuming that the extra DAC is to be the right channel, load INSPLAY and enter the following patch:

```
GET INSPLAY
SET 73E DF BF F3 BF D3
SAVE INSPLAYSTEREO 700 1FFF BA00 BBFF
```

This will create a new version of INSPLAY called INSPLAYSTEREO. No changes to the actual music object files are necessary for stereo operation. If the compiled songs did not specify a stereo pattern, voices 1 and 2 will be heard through the computer's DAC and voices 3 and 4 will be heard through the added DAC.

The following table lists the frequencies of all of the notes playable by INSPLAY in each of the standard temperaments along with their hex code when the pitch and octave offsets are zero. The frequencies shown are those actually generated by the program; not the ideal frequencies. All tunings are based on C8=4185.99Hz.

NOTE	HEX	FREQ TMPR=0	FREQ TMPR=1	FREQ TMPR=2	FREQ TMPR=3	FREQ TMPR=4
B		15.3926	15.3926	15.2587	15.2587	15.3926
C0		16.3295	16.3295	16.3295	16.3295	16.3295
C0#		17.2665	17.4003	17.1326	16.9988	17.2665
D0		18.3373	18.3373	18.0696	18.2034	18.3373
D0#		19.4081	19.2742	19.2742	19.5419	19.4081
E0		20.4788	20.6127	20.3450	20.3450	20.3450
F0		21.8173	21.6835	21.6835	21.8173	21.8173
F0#		23.0220	23.1558	22.8881	22.7543	23.0220
G0		24.4943	24.4943	24.4943	24.3605	24.4943
G0#		25.8328	26.1005	25.8328	25.4313	25.8328
A0		27.4390	27.5728	27.1713	27.3051	27.4390
A0#		29.0452	29.0452	29.0452	29.1790	29.0452
B0	01	30.7852	30.9191	30.6514	30.5175	30.7852
C1	02	32.6591	32.6591	32.6591	32.6591	32.6591
C1#	03	34.5330	34.8007	34.3991	34.1314	34.5330
D1	04	36.6746	36.6746	36.2730	36.5407	36.6746
D1#	05	38.8162	38.6823	38.6823	39.0839	38.8162
E1	06	41.0916	41.3593	40.8239	40.8239	40.8239
F1	07	43.6347	43.5009	43.5009	43.6347	43.6347
F1#	08	46.1779	46.4456	45.9102	45.6425	46.1779
G1	09	48.9887	48.9887	48.9887	48.8548	48.9887
G1#	0A	51.7995	52.3349	51.6657	50.9964	51.6657
A1	0B	54.8780	55.1457	54.4765	54.6103	54.8780
A1#	0C	58.2243	58.0904	58.0904	58.4920	58.2243
B1	0D	61.7043	61.9720	61.3028	61.0351	61.5705
C2	0E	65.3183	65.3183	65.3183	65.3183	65.3183
C2#	0F	69.1999	69.7353	68.9322	68.2629	69.0660
D2	10	73.3492	73.4831	72.5461	73.0815	73.4831
D2#	11	77.7662	77.4985	77.4985	78.1678	77.7662
E2	12	82.3171	82.7186	81.6478	81.6478	81.6478
F2	13	87.2695	87.1356	87.1356	87.4033	87.4033
F2#	14	92.4896	93.0250	91.9542	91.2850	92.4896
G2	15	97.9774	98.1113	98.1113	97.7097	98.1113
G2#	16	103.732	104.669	103.331	102.126	103.331
A2	17	109.890	110.291	108.953	109.354	109.890
A2#	18	116.448	116.180	116.180	116.984	116.582
B2	19	123.408	124.078	122.605	122.204	123.274
C3	1A	130.770	130.770	130.770	130.770	130.770
C3#	1B	138.533	139.604	137.864	136.659	138.266
D3	1C	146.698	147.100	145.226	146.163	147.100
D3#	1D	155.532	154.997	154.997	156.335	155.532
E3	1E	164.768	165.571	163.429	163.429	163.429
F3	1F	174.539	174.405	174.405	174.806	174.940
F3#	20	184.979	186.183	183.908	182.703	184.979
G3	21	195.954	196.222	196.222	195.553	196.222
G3#	22	207.599	209.473	206.662	204.387	206.662
A3	23	219.913	220.717	217.906	218.709	219.913
A3#	24	233.031	232.495	232.495	233.968	233.298
B3	25	246.817	248.289	245.211	244.408	246.549

C4	26	261.540	261.540	261.540	261.540	261.540
C4#	27	277.067	279.342	275.862	273.319	276.532
D4	28	293.530	294.333	290.586	292.460	294.333
D4#	29	311.065	309.994	309.994	312.805	311.065
E4	2A	329.536	331.142	326.859	326.859	326.859
F4	2B	349.212	348.810	348.810	349.747	350.015
F4#	2C	369.958	372.501	367.817	365.407	369.958
G4	2D	391.909	392.445	392.445	391.240	392.445
G4#	2E	415.199	419.081	413.325	408.774	413.325
A4	2F	439.961	441.434	435.946	437.418	439.961
A4#	30	466.062	464.991	464.991	467.936	466.597
B4	31	493.768	496.713	490.422	488.950	493.233
C5	32	523.215	523.215	523.215	523.215	523.215
C5#	33	554.268	558.819	551.725	546.639	553.064
D5	34	587.195	588.667	581.172	584.920	588.667
D5#	35	622.264	619.988	619.988	625.744	622.264
E5	36	659.206	662.284	653.852	653.852	653.852
F5	37	698.424	697.620	697.620	699.628	700.030
F5#	38	739.917	745.137	735.634	730.949	739.917
G5	39	783.953	784.890	784.890	782.615	784.890
G5#	3A	830.533	838.296	826.785	817.683	826.785
A5	3B	879.923	883.001	871.892	874.837	879.923
A5#	3C	932.258	930.116	930.116	936.006	933.329
B5	3D	987.671	993.427	980.845	978.034	986.600
C6	3E	1046.43	1046.43	1046.43	726.933	1046.43
C6#	3F	1108.67	1117.63	1103.58	1093.41	1106.12
D6		1174.52	1177.33	1162.47	1169.84	1177.33
D6#		1244.52	1240.11	1240.11	1251.62	1244.52
E6		1318.41	1324.56	1307.83	1307.83	1307.83
F6		1396.84	1395.24	1395.24	1399.25	1400.06
F6#		1479.96	1490.27	1471.40	1462.03	1479.96
G6		1567.90	1569.78	1569.78	1565.23	1569.78
G6#		1661.19	1676.59	1653.57	1635.50	1653.57
A6		1759.98	1766.13	1743.78	1749.80	1759.98
A6#		1864.65	1860.36	1860.36	1872.14	1866.79
B6		1975.47	1986.98	1961.82	1956.20	1973.20
C7		2092.99	2092.99	2092.99	2092.99	2092.99
C7#		2217.47	2235.41	2207.17	2186.95	2212.39
D7		2349.18	2354.80	2324.95	2339.81	2354.80
D7#		2489.05	2480.35	2480.35	2503.37	2489.05
E7		2636.95	2649.27	2615.81	2615.81	2615.81
F7		2793.83	2790.61	2790.61	2798.64	2800.25
F7#		2959.93	2980.54	2942.93	2924.19	2959.93
G7		3135.94	3139.56	3139.56	3130.46	3139.56
G7#		3322.39	3353.31	3307.14	3271.00	3307.14
A7		3519.96	3532.27	3487.56	3499.74	3519.96
A7#		3729.30	3720.73	3720.73	3744.42	3733.58
B7		3951.08	3973.97	3923.78	3912.40	3946.53
C8		4185.99	4185.99	4185.99	4185.99	4185.99

HIGHEST USABLE NOTE VS HARMONIC NUMBER

HIGHEST HARMONIC	HIGHEST NOTE 3500HZ LIMIT	HIGHEST HARMONIC	HIGHEST NOTE 3500HZ LIMIT
1	G#7	17	G3
2	G#6	18	F#3
3	C#6	19	F3
4	G#5	20	F3
5	F5	21	E3
6	C#5	22	D#3
7	B4	23	D3
8	G#4	24	C#3
9	F#4	25	C#3
10	F4	26	C3
11	D#4	27	B2
12	C#4	28	B2
13	C4	29	A#2
14	B3	30	A#2
15	A#3	31	A2
16	G#3	32	A2

INSMUS-8 MANUAL REV. A ERRATA

Please mark the following changes in your INSMUS-8 manual:

NEWINS and WAVESET Statements

It is recommended that the strongest harmonic(s) in these statements be given an amplitude of 100% and the less strong ones be given lesser amplitudes as desired. If this is not done, a warning message may be displayed on the console and the final "optimized" amplitude may be less than maximum.

SEQTAB Statement

The waveform numbers used in this statement start at ONE and go up through N. The example in section 4.3.2.1 should read:

```
SEQTAB  3  X=0  BR=01  W2:1,16
```

Similarly, the first example in section 4.3.2.2 should read:

```
SEQTAB  4  X=9  BR=10 W4:2,10   BR=21 W5:1,8
```

and the second example in section 4.3.2.2 should read:

```
SEQTAB  4  X=8   BR=10 W4:2,10   BR=21 W5:1,8
```

Note that the INSNOTRAN compiler will not flag as an error the use of waveform number 0 in a SEQTAB statement. The resulting sequence table however will refer to a spurious waveform for number zero which usually results in a "chink" in the instrument sound.